jQuery Mobile
快速入门

[美] Brad Broulik 著
巩亚萍 姚婷 译

前言

当前，企业和个人用于开发和发布移动应用程序所使用的技术正在发生变化，而我们则是见证者。最初，开发和发布移动程序的策略是针对每一个主流平台开发独立的本地 app。然而，开发团队迅速意识到，维护多个平台所需的花费是不可忍受的，而且移动团队也会丧失其敏捷性。在将来，移动开发团队只需一次编码，就可以将 app 部署到所有设备上，这样的开发团队会更具竞争性，而 jQuery Mobile 可以帮助你实现这一目标。

jQuery Mobile 是一个框架，用于交付具有统一界面的跨平台移动 Web 应用程序。jQuery Mobile 将响应式布局与渐进式增强结合起来，从而通过一个代码库来呈现最佳的用户体验。通过使用 jQuery Mobile，我们将会看到如何为 iOS、Android、Windows Phone、BlackBerry 及其他移动设备创建可主题化的，而且具有本地外观的相应式应用程序。我们会发现是什么让 jQuery Mobile 不同于其他移动 Web 开发平台，也会通过示例来探索 jQuery Mobile 的特性，其中包括设计元素和事件处理。

主要内容

- jQuery Mobile 的独特特性；
- jQuery Mobile 的核心特性，其中包括页面结构、导航、表单元素、列表和表格；
- 如何创建可主题化的设计；
- 完整的 jQuery Mobile API，其中包括数据属性、方法和事件；
- 将 Web 服务、Google Maps 和地理定位集成到 jQuery Mobile app 中；
- 当需要将 app 发布到 app store 或者需要访问设备功能时，如何使用 PhoneGap 来扩展 jQuery Mobile；

目录

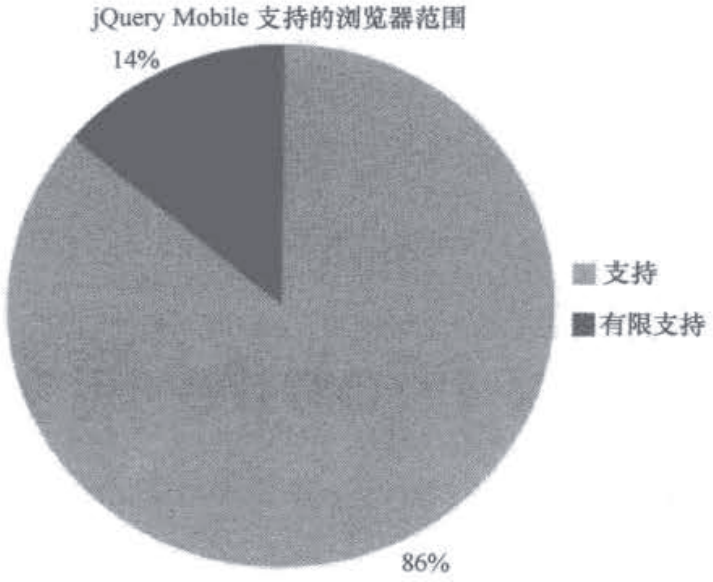
第 1 章

jQuery Mobile 简介

jQuery Mobile 是一个用来构建跨平台移动 Web 应用程序的新 UI 框架，具有使用简单的特点。在短短几分钟内，你可以创建出能够在当今可用的近乎所有手机、平板电脑、台式机和电子阅读器设备上运行的优化移动应用程序（app）。是的，通过一个 jQuery Mobile 代码库，我们几乎可以为所有的消费者创建统一标准的体验。当 Web 设计人员或开发人员需要一个简单的框架来开发丰富的移动 Web 体验时，jQuery Mobile 就成为理想之选。而且这个丰富的移动体验并不局限于 Web。jQuery Mobile app 也可以使用混合技术来编译，从而将它分发在你最喜欢的本地 app 商店。现在开启我们的 jQuery Mobile 之旅，我们首先看一下可以体现 jQuery Mobile 独特性的那些重要特性。

1.1 通用访问

带有浏览器的所有设备都可以访问 jQuery Mobile 应用程序。这对 jQuery Mobile 的分发模型（见图 1-1）而言，是一个有利的优势。几乎所有的移动设备在出厂时都自带一个浏览器。如果你的 app 可以让近乎所有的移动设备来访问，这将使得它极具竞争力。下面是 jQuery Mobile 1.0 所支持的一个完整的设备列表，其中包括大多数手机、平板电脑、桌面浏览器，甚至是电子阅读器。



http://gs.statcounter.com/#mobile_browser-ww-monthly-201101-201106-bar

图 1-1 jQuery Mobile 1.0 支持的浏览器范围

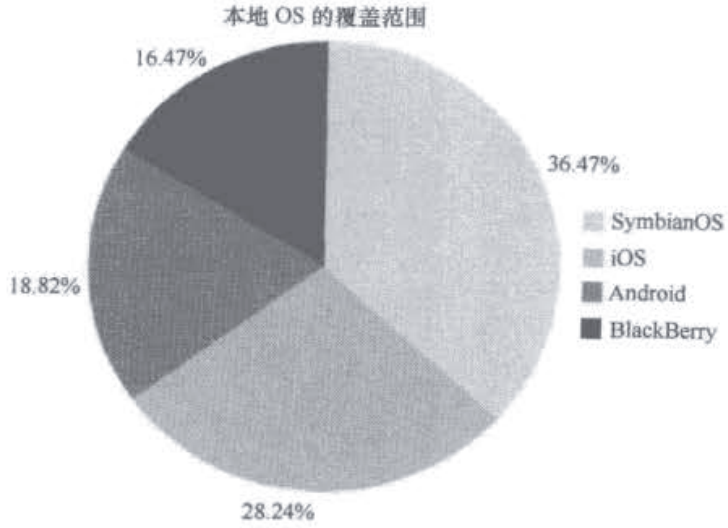
支持的设备

- 手机/平板电脑
  - Android 1.6+
  - BlackBerry 5+
  - iOS 3+
  - Windows Phone 7
  - WebOS 1.4+
  - Symbian（Nokia S60）
  - Firefox Mobile Opera Mobile 11+
  - Opera Mini 5+
- 桌面浏览器
  - Chrome 11+
  - Firefox 3.6+
  - Internet Explorer 7+

- Safari
- 电子阅读器
  - Kindle
  - Nook

> **注意**：有关所有支持平台的最新列表，请参考 jQuery Mobile 的支持平台页面（链接为 http://jquerymobile.com/gbs/）。

相对而言，本地应用程序的开发有一个非常严格的分发模型（见图 1-2）。本地应用程序只能用于它们本地的操作系统。例如，一个 iPhone 的 app 只能通过 iOS 设备来访问。如果你希望开发的应用程序可以被所有的消费者访问，则这种分发模型就无能为力了。幸运的是，使用 jQuery Mobile 开发的 app 则不受这一分发障碍的限制。



http://gs.statcounter.com/#mobile_os-ww-monthly-201101-201106-bar

图 1-2 本地 OS 的覆盖范围

除了通用访问的特性之外，jQuery Mobile 应用程序也可以利用我们在 Web 上已经相当习惯的及时部署能力。对 jQuery Mobile app 来说，关于本地 app 分发模型所需要的认证审查（certification review）方面，也不存在障碍。移动 Web app 可以立即更新并部署到你的生产用户（production user）上。例如，我最近正在开发一个需要进行一次更新的本地企业应用程序，为了核准这一更新，它需要花费一周的时间用于重新认证过程。平心而论，本地 app 商店也有可以提交紧急更新的选项，但问题是你需要依赖第三方将更新推送到他们的商店。在这一方面，移动 Web 的即时部署模型相当具有优势。

1.2 跨所有移动平台的统一 UI

通过采用 HTML5 和 CSS3 标准，jQuery Mobile 提供了一个统一的用户界面（User Interface，UI）。移动用户希望他们的用户体验能够在所有平台上保持一致（见图 1-3、图 1-4 和图 1-5）。然而，通过比较 iPhone 和 Android 上的本地 Twitter app，即可发现用户体验并不统一。jQuery Mobile 应用程序解决了这种不一致性，提供给用户一个与平台无关的用户体验，而这正是用户熟悉和期待的。此外，统一的用户界面还会提供一致的文档、屏幕截图和培训，而不管终端用户使用的是什么平台。例如，如果你的销售团队需要一个与即将部署的新的移动 app 相关的培训，则用户文档会包含一致的可以应用到所有平台的屏幕截图。如果团队中有半数使用的是 iPhone，另外一半使用的是 Android 设备，对他们所有人来说，培训体验和文档是相同的。

图 1-3 iPhone

图 1-4 Windows Phone

图 1-5 Android

jQuery Mobile 也有助于消除为特定设备自定义 UI 的需求。一个 jQuery Mobile 代码库可以在所有支持的平台上呈现出一致性，而且无需进行自定义。与为每个

OS 提供一个本地代码库的组织结构相比，这是一种费用非常低廉的解决方案。而且就支持和维护成本而言，从长远来看支持一个单一的代码库也颇具成本效益（见图 1-6）。

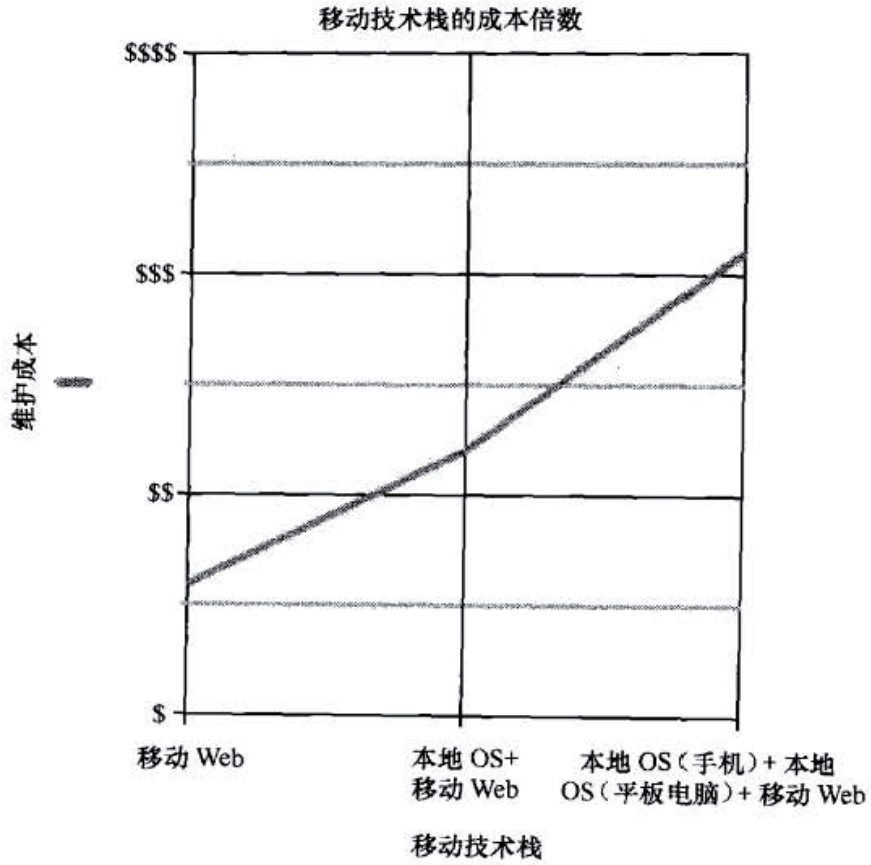


图 1-6 移动技术栈的成本倍数

1.3 简化的标记驱动的开发

jQuery Mobile 页面是使用 HTML 5 标记设计（styled）的（见程序清单 1-1）。除了 HTML5 中新引入的自定义数据属性之外，其他一切东西对 Web 设计人员和开发人员来讲都很熟悉。如果你已经很熟悉 HTML5，则转移到 jQuery Mobile 也应算是一个相对无缝的转换。就 JavaScript 和 CSS 而言，jQuery Mobile 在默认情况下承担了所有负担。但是，有些情况下，仍然需要依赖 JavaScript 来创建更为动态的或增强的页面体验。除了设计页面时用到的标记具有简洁性之外，jQuery Mobile 还可以迅速地原型化用户界面。我们可以迅速创建功能页面、转换和插件（widget）的静态工作流，从而通过最少的付出让用户看到活生生的原型。

程序清单 1-1 使用 HTML 标记设计的页面

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Title</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="jquery.mobile-1.0.min.css" />
  <script type="text/javascript" src="jquery-1.6.2.min.js"></script>
  <script type="text/javascript" src="jquery.mobile-1.0.min.js"></script>
</head>
<body>

<div data-role="page">
    <div data-role="header">
        <h1>Page Header</h1>
    </div>

    <div data-role="content">
        <p>Hello jQuery Mobile!</p>
    </div>

    <div data-role="footer">
        <h4>Page Footer</h4>
    </div>

</div>

</body>
</html>
```

1.4 渐进式增强

jQuery Mobile 可以为一个设备呈现出可能是最优雅的用户体验。例如，我们来看图 1-7 中的 jQuery Mobile 开关控件。这是 A 级（A-Grade）浏览器[1]上的一个开关控件。

jQuery Mobile 可以呈现出应用了完整 CSS3 样式的控件。而图 1-8 是同一个开关控件，只不过它是在一个比较老的 C 级浏览器[2]上呈现出来的。C 级浏览器不能呈现完整的 CSS3 样式[2]。

> **重要：**尽管从视觉上来讲，C 级的体验并不是最吸引人的，但是它可以演示平稳降级的有效性。随着用户升级到较新的设备，C 级浏览器市场最终会减小。但是在 C 级浏览器退出市场之前，当运行 jQuery Mobile app 时，仍然可以得到实用的用户体验。

[1] A 级浏览器支持媒体查询，而且可以从 jQuery Mobile CSS 3 样式（styling）中呈现出可能是最佳的体验。
[2] C 级浏览器不支持媒体查询，也无法从 jQuery Mobile 中接收样式增强。

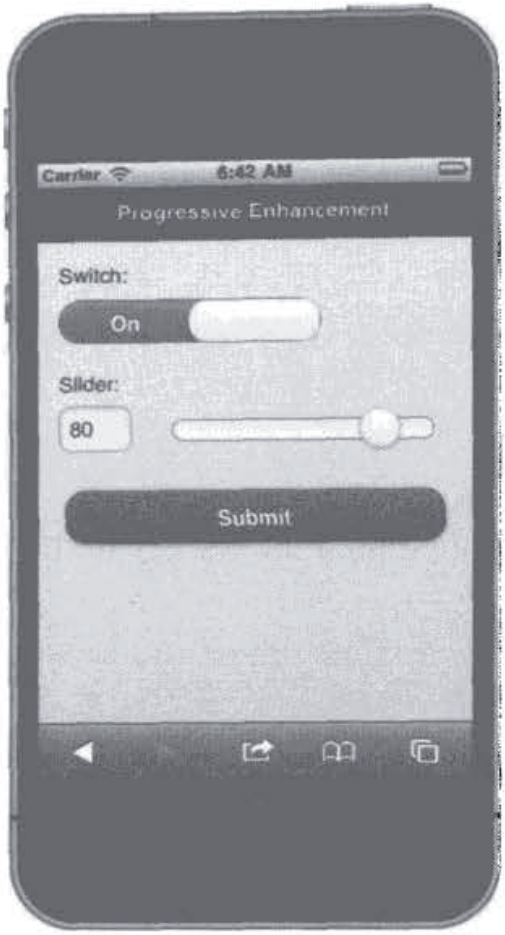


图 1-7 A 级体验

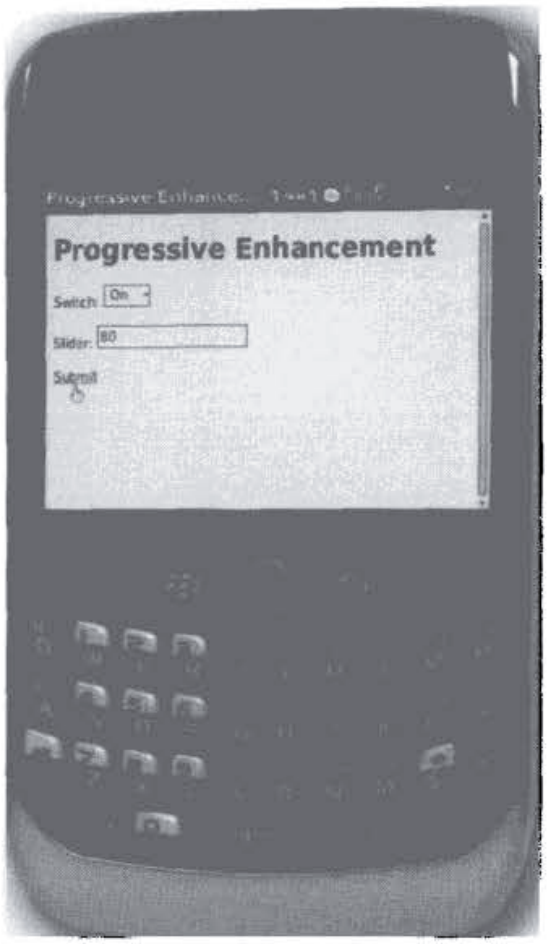


图 1-8 C 级体验

本地应用程序并不能总是平稳地降级。在大多数情况下，如果你的设备不支持本地 app 特性（feature），甚至不能下载 app。例如，iOS 5 中的一个新特性是 iCloud 存储，这个新特性使多个设备间的数据同步更为简化。出于兼容性考虑，如果创建了一个包含这个新特性的 iOS app，则需要将 app 的“minimum allowed SDK”（允许的最低 SDK）设置为 5.0。当我们的 app 出现在 App Store 中时，只有运行 iOS 5.0 或者更高版本的用户才能看到。在这一方面，jQuery Mobile 应用程序更具灵活性。

1.5 响应式设计

jQuery Mobile UI 可以根据不同的显示尺寸来呈现。例如，同一个 UI 会恰如其分地显示在手机（见图 1-9）或更大的设备上，比如平板电脑、台式机或电视（见图 1-10）。

图 1-9 手机显示

图 1-10 平板电脑/台式机/电视显示

1.5.1 一次构建，随处运行

有没有可能构建一个可用于所有消费者（手机、台式机和平板电脑）的应用程序呢？是的，这完全有可能。Web 提供了一个通用的分发方式。jQuery Mobile 提供了跨浏览器的支持。例如，在较小的设备上我们可以使用带有简要内容的小图片，而在较大的设备上我们则可以使用带有详细内容的较大图片。如今，具有移动呈现功能（mobile presence）的大多数系统通常都支持桌面式 Web 和移动站点。在任何时候，只要你必须支持一个应用程序的多个分发版本，就会造成浪费。系统根据自己的需要“支持”移动呈现，以避免浪费的速率，会促成“一次构建、随处运行”的神话得以实现。

响应式表单

在某些情况下，jQuery Mobile 可以为用户创建响应式设计。下面将讲解 jQuery Mobile 的响应式设计如何良好地应用于竖屏（portrait）模式和横屏（landscape）模式中的表单字段。例如，在竖屏视图中（见图 1-11），标签位于表单字段的上面。而当将设备横屏放置时（见图 1-12），表单字段和标签并排显示。这种响应式设计可以基于设

图 1-11 响应式设计（竖屏）

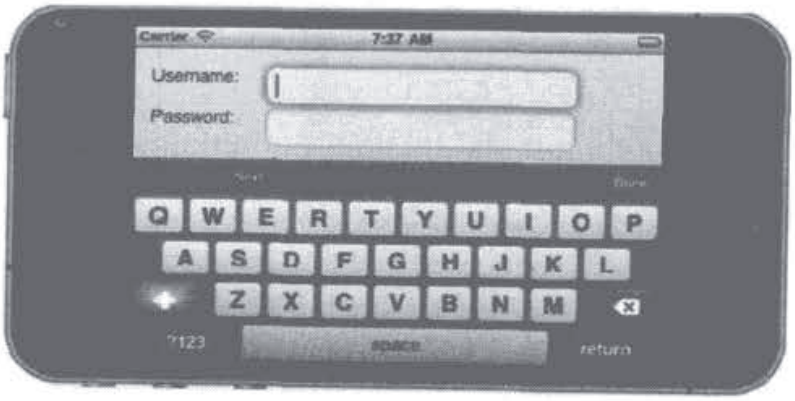


图 1-12 响应式设计（横屏）

备可用的屏幕真实状态提供最合用的体验。jQuery Mobile 为用户提供了很多这样优秀的 UX（用户体验）操作方法，而且不需要用户付出半分力气。

1.6 可主题化的设计

jQuery Mobile 提供另一个可主题化的设计，它可以允许设计人员快速地重新设计他们的 UI。默认情况下，jQuery Mobile 提供了 5 个可主题化的设计，而且可以灵活地互换所有组件的主题，其中包括页面、标题、内容和页脚组件。创建自定义主题的最有用的工具是 ThemeRoller[3]。

我们可以轻易地重新设计一个 UI。例如，我可以迅速采用 jQuery Mobile 应用程序一个默认的主题（见图 1-13），然后在几秒钟时间内就可以使用另外一个内置的主题来重

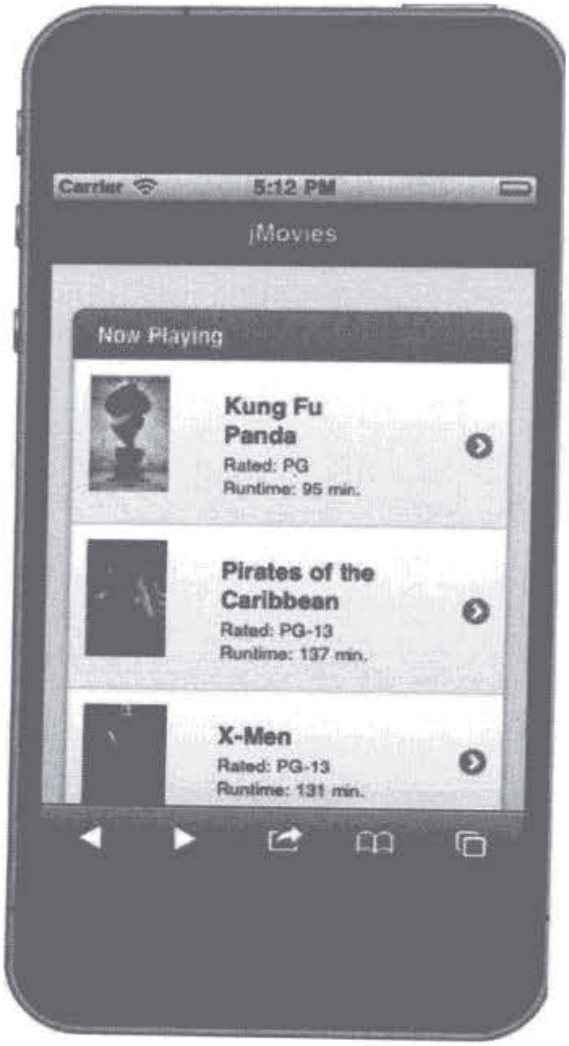


图 1-13 默认主题

[3] 请参见 http://jqueryui.com/themeroller/。ThemeRoller 是一个基于 Web 的工具，它可以将生成基于 CSS 的新主题的过程进行自动化操作。

新设计默认主题。在修改主题时（见图 1-14），我从列表中选择了另外一个主题。唯一需要添加的一个标记是 data-theme 属性。我们将在第 7 章详细讨论关于主题的相关知识。

```
<!-- Set the lists background to black -->
<ul data-role="listview" data-inset="true" data-theme="a">
```

图 1-14 选择的另外一个主题

1.7 可访问性

jQuery Mobile app 在默认情况下是 508 兼容的，这是一个对任何人来说都很有价值的特点[4]。

尤其是政府或国家机构要求他们的应用程序必须是 100%可以访问的。而且，移动

[4] 508 兼容是一项联邦规则，它要求应用程序必须可以让残疾人用户来访问。移动 Web 上最常使用的辅助技术是屏幕阅读器。

屏幕阅读器的使用量正在逐年增长。据 WebAIM[5] 报道，66.7%的屏幕阅读器用户都在他们的移动设备上使用屏幕阅读器。

> **注意：**如果想知道你的移动站点是否是 508 兼容的，可以使用 WAVE[6] 来进行评估。

除了使用 WAVE 来测试的你移动 app 的可访问性之外，通过使用真实的辅助技术来实际测试你的移动 Web 应用程序，也是很有价值的。例如，如果你有一台 iOS 设备，激活 Apple 的辅助工具 VoiceOver[7]，并率先体验这一测试行为。

> **注意：**如果你有兴趣查看现有的 jQuery Mobile 应用程序，可以查看在线 jQuery Mobile Gallary（地址为 http://www.jqmgallery.com/），它可以激发你的想法和灵感。

1.8 总结

在本章，我们讲解了一些 jQuery Mobile 独一无二的重要特性。

- 带有浏览器的所有设备都可以访问 jQuery Mobile app，而且这些 app 在优化之后可以在当今几乎所有可用的手机、平板电脑、台式机，以及电子阅读器设备上运行。

- jQuery Mobile 应用程序也可以利用我们在 Web 上已经相当习惯的及时部署能力。

- 一个 jQuery Mobile 代码库可以在所有可以支持的平台上呈现出一致性，而且无需进行自定义。与为每一个 OS 或客户端构建 app 相比，这是一种费用相当低廉的解决方案。

- jQuery Mobile 是一个简化的标记驱动的框架，Web 设计人员和开发人员对这样的框架并不陌生。但是，你可以用 100%的标记来构建 jQuery Mobile app，面对这样的现实，我们除了惊讶就是兴奋。

---

5 见 http://webaim.org/projects/screenreadersurvey3/#mobileusage.
6 见 http://wave.webaim.org/.
7 见 http://www.apple.com/accessibility/iphoen/vision.html.

■ jQuery Mobile 利用日渐增强的技术为所有的 A 级设备提供了非常丰富的用户体验，同时为较老的 C 级浏览器提供了非常实用的用户体验。

■ jQuery Mobile UI 可以根据设备的不同尺寸进行呈现，其中包括电话、平板电脑、台式机或电视。

■ jQuery Mobile 支持主题化的设计，这使得设计人员能够在全局快速地重新设计他们的 UI。

■ 所有的 jQuery Mobile 应用程序都是 508 兼容的。

第2章

jQuery Mobile 入门

在第 1 章，我们讲解了令 jQuery Mobile 独一无二的一些重要特征，现在我们开始讲解 jQuery Mobile 的基础知识，以便我们迅速上手。我们首先来概览 jQuery Mobile 页面模板。我们可以实际选择的页面模板有两个，后面将讨论每一个的优势。然后，我们会讲解 jQuery Mobile 是如何将我们的语义标记提升转化为一个优化的移动体验的。此外，我们还会介绍 jQuery Mobile 导航模型的工作方式。尽管 jQuery Mobile 能够管理整个导航体验，但是理解导航模式的工作方式仍然很重要。最后，我们会讲解如何真正让你的页面转换广受欢迎。迫不及待地想要赶紧开始了吧？我们先来看一个 jQuery Mobile 页面的示例。

2.1 jQuery Mobile 页面模板

一个 jQuery Mobile 页面模板如程序清单 2-1 所示。在往下讲解之前，我们先来运行一下这个模板。复制 HTML 模板（ch2/template.html），然后粘贴到你的台式机中，并通过你最喜欢的浏览器中打开它。现在，你运行的就是一个 jQuery Mobile app，而且无论使用的是什么浏览器，它看起来应该与图 2-1 所示相同。该模板符合 HTML5 语法标准，并且包含了 jQuery Mobile 的特定属性和 asset 文件（CSS、js）。在程序清单 2-1 中，每一个特定的 jQuery Mobile asset 和属性都突出显示，并进行了解释。

程序清单 2-1 jQuery Mobile 页面模板（ch2/template.html）

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Title</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="jquery.mobile.css" />
    <script type="text/javascript" src="jquery.js"></script>
     <!--<script src="custom-scripts-here.js"></script>-->
     <script type="text/javascript" src="jquery.mobile.js"></script>
</head>
<body>

<div data-role="page">
    <div data-role="header">
        <h1>Page Header</h1>
    </div>

    <div data-role="content">
         <p>Hello jQuery Mobile!</p>
    </div>

    <div data-role="footer">
        <h4>Page Footer</h4>
    </div>
</div>

</body>
</html>
```

1．对 jQuery Mobile 来说，这是一个推荐的视图（viewport）配置。device-width 值表示，我们希望让内容扩展到屏幕的整个宽度。initial-scale 设置了用来查看 Web 页面的初始缩放百分比或缩放因数。值为 1，则显示一个未缩放的文档。作为一名 jQuery Mobile 开发人员，你可以根据应用程序的需要自定义视图的设置。例如，如果你希望禁用缩放，则可以添加 user-scalable=no。然而，如果禁用了缩放，则会破坏应用程序的可访问性，因此要谨慎使用。

2．jQuery Mobile 的 CSS 会为所有的 A 级和 B 级浏览器应用风格（stylistic）的优化。你可以根据需要自定义或添加自己的 CSS。

3．jQuery 库是 jQuery Mobile 的一个核心依赖，如果你的 app 需要更多动态行为，则强烈建议在你的移动页面中使用 jQuery 的核心 API。

4．如果你需要改写 jQuery Mobile 的默认配置，则可以应用你的自定义设置。有

关自定义 jQuery Mobile 默认配置的细节，请参考第 8 章。

5．jQuery Mobile JavaScript 库必须在 jQuery 和任何可能存在的自定义脚本之后声明。jQuery Mobile 库是增强整个移动体验的核心。

6．data-role=“page”为一个 jQuery Mobile 页面定义了页面容器。只有在构建多页面设计时，才会用到该元素（见程序清单 2-3）。

7．data-role=“header”是页眉（header）或标题栏，如图 2-1 所示。该属性是可选的。

8．data-role=“content”是内容主体的包装容器（wrapping container）。该属性是可选的。

9．data-role=“footer”包含页脚栏（见图 2-1）。该属性是可选的。

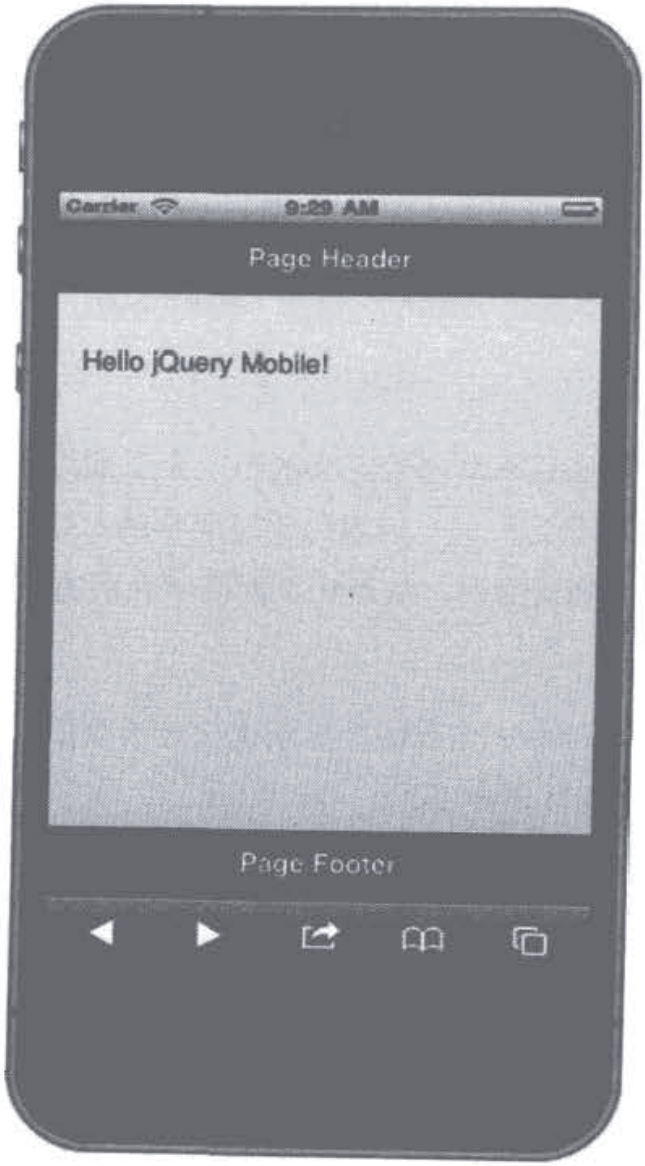


图 2–1 Hello jQuery Mobile

重要：CSS 和 JavaScript 文件的顺序必须与程序清单 2-1 中列出的顺序一致。这种排序是必要的，它可以正确地初始化这些文件，然后再让 jQuery Mobile 来引用。此外，建议从内容分发网络（Content Delivery Network，CDN）下载这些文件。尤其是，你可以从 jQuery Mobile CDN[1] 下载这些文件。这些从 CDN 中下载的文件都是经过高度优化的，可以为用户提供更好的响应式体验。它们是一些缓存的压缩文件，体积很小，因此可以并行载入。

提示：为了将页脚定位到屏幕的最底部，可以为页脚元素添加 data-position="fixed"。默认的页脚位置是在内容之后，并不是屏幕的底部。例如，如果你的内容只占据了一半的屏幕高度，则页脚会出现在屏幕中间。

```
<div data-role="footer" data-postion="fixed">
```

2.1.1 jQuery Mobile 页面增强

jQuery Mobile 是如何为优化的移动体验增强标记的呢？我们来看图 2-2。

1. 首先，jQuery Mobile 会载入语义 HTML 标记（见程序清单 2-1）。

2. 其次，jQuery Mobile 会迭代由它们的 data-role 属性定义的每一个页面组件。由于 jQuery Mobile 迭代每一个页面组件，因此会为每一个应用优化过的移动 CSS3 组件添加标记。jQuery Mobile 最终会将标记添加到页面中，从而让页面能够在所有平台上普遍呈现。

3. 最后，在完成页面的标记添加之后，jQuery Mobile 会显示优化过的页面。要查看由移动浏览器呈现的添加源文件，请参考程序清单 2-2。

[1] 见 http://jquerymobile.com/download/.

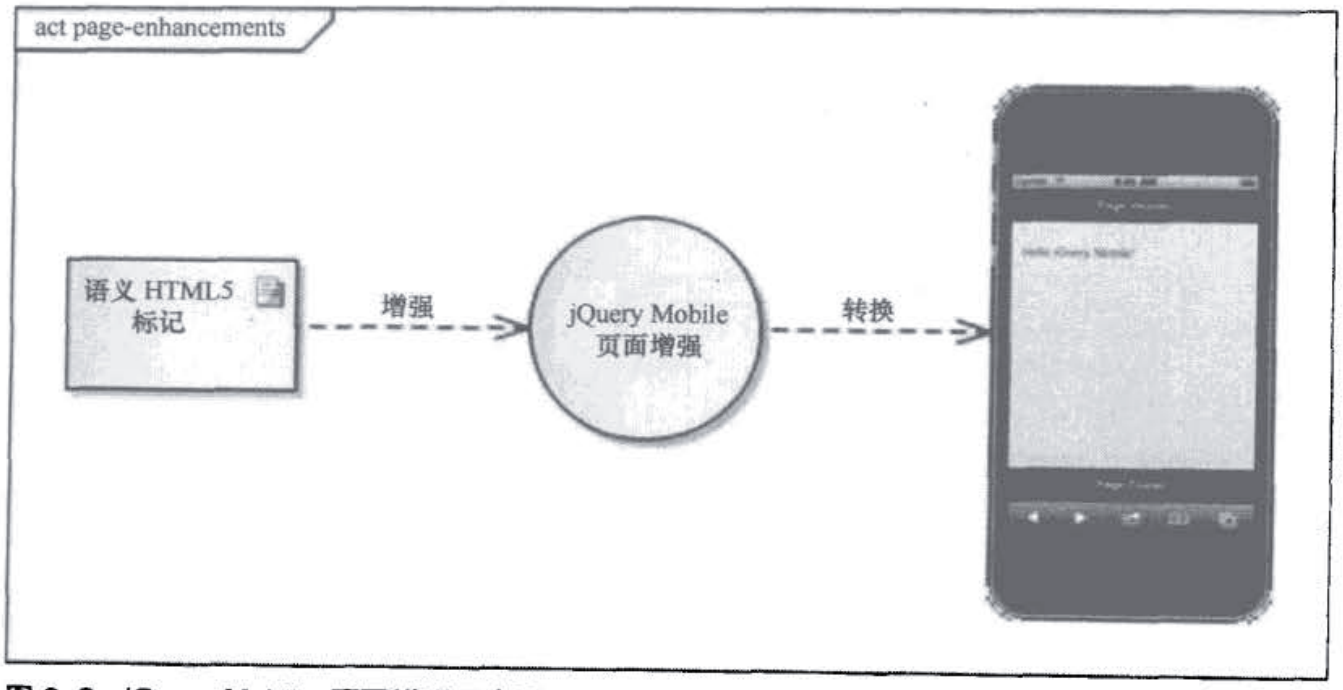


图 2-2 jQuery Mobile 页面增强示意图

程序清单 2-2 jQuery Mobile 增强的 DOM

```
<!DOCTYPE html>
<html class="ui-mobile">
 <head>
   <base href="http://www.server.com/app-name/path/">                    1
   <meta charset="utf-8">
   <title>Page Header</title>
   <meta content="width=device-width, initial-scale=1" name="viewport">
   <link rel="stylesheet" type="text/css" href="jquery.mobile-min.css" />
    <script type="text/javascript" src="jquery-min.js"></script>
    <script type="text/javascript" src="jquery.mobile-min.js"></script>
 </head>

<body class="ui-mobile-viewport">                                        2
   <div class="ui-page ui-body-c ui-page-active" data-role="page"
         style="min-height: 320px;">
     <div class="ui-bar-a ui-header" data-role="header" role="banner">
       <h1 class="ui-title" tabindex="0" role="heading" aria-level="1">
           Page Header
       </h1>
     </div>

     <div class="ui-content" data-role="content" role="main">
```

```
        <p>Hello jQuery Mobile!</p>
      </div>

      <div class="ui-bar-a ui-footer ui-footer-fixed fade ui-fixed-inline"
           data-position="fixed" data-role="footer" role="contentinfo"
           style="top: 508px;">
        <h4 class="ui-title" tabindex="0" role="heading" aria-level="1">
          Page Footer
        </h4>
      </div>
    </div>

    <div class="ui-loader ui-body-a ui-corner-all" style="top: 334.5px;">
      <span class="ui-icon ui-icon-loading spin"></span>
      <h1>loading</h1>
    </div>

  </body>
</html>
```

1. base 标签（tag）的@href 为一个页面中的所有链接指定了一个默认的地址或者默认的目标。例如，当载入特定页面的资源（assets）时（比如图片、CSS、js 等）jQuery Mobile 会用到@href。

2. body 标签包含了 header、content 和 footer 组件的增强样式。默认情况下，所有的组件都是使用默认的主题和特定的移动 CSS 增强来设计（styled）的。作为一个额外的好处，所有的组件现在都证明了可访问性，而这要归功于 WAI-ARIA 角色和级别。我们可以免费获得这些增强。

现在你应该感觉到，可以很容易地设计一个基本的 jQuery Mobile 页面了。我们前面已经介绍了核心的页面组件（page、header、content、footer），并看到了一个增强的 jQuery Mobile 页面所产生的文档对象模型（Document Object Model, DOM）。接下来，我们开始讲解 jQuery Mobile 的多页面模板。

2.2 多页面模板

jQuery Mobile 支持在一个 HTML 文档中嵌入多个页面的能力，如程序清单 2-3 所示。该策略可以用来预先获取最前面的多个页面，当载入子页面时，其响应时间会缩短。在下面的例子中可以看到，多页面文档与我们前面看到的单页面文档相同，第二个页面附加在第一个页面后面的情况除外。多页面的具体细节在程序清单 2-3 中有突出显示，并会在下面进行讲解。

程序清单 2-3 多页面模板

```
<head>
  <meta charset="utf-8">
  <title>Multi Page Example</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" href="jquery.mobile-min.css" />
  <script type="text/javascript" src="jquery-min.js"></script>
  <script type="text/javascript">
    /* Shared scripts for all internal and ajax-loaded pages */
  </script>
  <script type="text/javascript" src="jquery.mobile-min.js"></script>
</head>

<body>

<!-- First Page -->
<div data-role="page" id="home" data-title="Home">                    1
    <div data-role="header">
        <h1>Welcome Home</h1>
    </div>

    <div data-role="content">
        <a href="#contact" data-role="button">Contact Us</a>          2
    </div>
</div>

<!-- Second Page -->
<div data-role="page" id="contact" data-title="Contact">
    <div data-role="header">
        <h1>Contact Us</h1>
    </div>

    <div data-role="content">
        Contact information...
    </div>
    <script type="text/javascript">
        /* Page specific scripts here */                              3
    </script>
</div>

</body>
```

1. 多页面文档中的每一个页面必须包含一个唯一的 id。每个页面可以有一个 page 或 dialog 的 data-role。最初显示多页面时，只有第一个页面得到了增强并显示出来。例如，当请求 multi-page.html 文档时，其 id 为“home”的页面将会显示出来，原因是它是多页面文档中的第一个页面。如果想要请求 id 为“contact”的页面，则可以通过在多页面文档名的后面添加#号，以内部页面的 id 名方式来显示，此时就是 multi-page.html#contact。当载入一个多页面文档时，只有初始页面会被增强并显示。后续页面只有当被请求并被缓存到 DOM 内时，才会被增强。对于要求有快速响应时间的页面来说，该行为是很理想的。为了设置每一个内部页面的标题，可以添加 data-title 属性。

2．当链接到一个内部页面时，必须通过页面的 id 来引用。例如，contact 页面的 href 链接必须被设置为 href=“#contact”。

3．如果你想查看特定页面中的脚本，则它们必须被放置在页面容器内。该规则同样也适用于通过 Ajax 载入的页面（我们将在下一节详细介绍）。例如，在 multi-page.html#contact 的内部声明的任何 JavaScript 无法被 multi-page.html#home 来访问。只有活跃页面的脚本可以被访问。但是，在父文档的 head 标签内声明的所有的脚本，包括 jQuery、jQuery Mobile 和你自己的自定义脚本，都可以被内部页面和通过 Ajax 载入的页面来访问。

2.2.1 设置内部页面的页面标题

需要重点注意的是，内部页面的标题（title）可以按照如下优先顺序进行设置。

1．如果 data-title 值存在，则它会用作有内部页面的标题。例如，“multi-page.ht ml#home”页面的标题将被设置为“Home”。

2．如果不存在 data-title 值，则页眉（header）将会用作内部页面的标题。例如，如果“multi-page.html#home”页面的 data-title 属性不存在，则标题将被设置为页面 header 标记的值“Welcome Home”。

3．最后，如果内部页面既不存在 data-title，也不存在页眉，则 head 标记中的 title 元素将会用作内部页面的标题。例如，如果“multi-page.html#page”页面不存在 data-title 属性，也不存在页眉，则该页面的标题将被设置为其父文档的 title 标记的值“Multi Page Example”。

重要：当链接到一个包含多个页面的页面时，必须为其链接添加 rel=“external”。

```
<!-- Must include rel="external" when linking to multi-page documents -->
<a href="multi-page.html" rel="external">Home</a>

<!-- May optionally use the target attribute -->
<a href="multi-page.html" target="_blank">Home</a>
```

这将会执行一个全页面的刷新。由于 jQuery Mobile 无法将一个多页面文档载入到活动页面的 DOM 中，因此会用到这个刷新。jQuery Mobile 之所以无法将多页面文档载入到 DOM 中，是因为它使用了 URL（#），由此产生了命名空间冲突。

jQuery Mobile 使用#来识别多页面文档中的内部页面。

此外，由于 jQuery Mobile 使用了#来识别 DOM 内的唯一页面，因此无法使用锚标签书签特性（index.html#my-bookmark）。jQuery Mobile 会将 my-bookmark 当作一个页面识别符，而不是当作书签。Ajax 驱动的导航将在下面的小节进行详细讲解。

2.2.2 单页面文档与多页面文档的对比

你需要确定页面访问的发展趋势，以方便从带宽和响应时间的角度来选择最合适的广式。多页面文档在最初载入时，会占用较多的带宽，但是只需要向服务器发送一个请求即可，因此它们的子页面会以相当短的响应时间载入。而单页面文档尽管占用的带宽较少，但是每访问一个页面，就需要向服务器发送一个请求，因此响应时间会比较长。

如果你通常会按顺序访问多个页面，则最为理想的方式是将它们放置在同一个文档内的最前面，以方便载入。这样尽管最初占用的带宽会略高，但是在访问下一个页面时，可以实现即时响应。如果用户同时访问两个页面（尽管概率很低，但毕竟存在），则可以将文件单独存放，从而在初次载入时能够消耗较少的带宽。

现在有一些可用的工具，可以辅助收集页面访问趋势或者其他度量，从而帮助优化页面访问方式。例如，Google Analytics[2] 或 Omniture[3] 都是常见的用于分析移动 Web 应用程序的解决方案。

提示：在大多数情况下，建议使用单页面模型，然后在后台将常用的页面动态添加到 DOM 中。通过为我们希望动态载入的任何链接添加 data-prefetch 属性，即可实现该行为。

```
<a href="load-page-dynamically.html" data-prefetch></a>
```

这种混合方式可以让我们有选择性地选择想要载入和缓存的链接。再次强调，只建议将该方法用于需要频繁访问的页面，原因是该行为会引发一个额外的 HTTP 请求，以动态载入页面。

2 见 http://www.google.com/analytics/.
3 见 http://www.omniture.com/.

2.3 Ajax 驱动的导航

在上面提到的多页面案例中（见程序清单 2-3），我们了解到 jQuery Mobile 如何从一个内部页面导航到另外一个内部页面。当多页面文档在初始化时，内部页面已经添加到 DOM 中，这样从一个内部页面转换到另外一个页面时，速度才会相当快。在从一个页面导航到另外一个页面时，我们可以配置要应用的页面转换类型。默认情况下，框架会为所有的转换应用一个“滑动（slide）”效果。在本章后面，我们会讨论可以选择的转换和转换类型。

```
<!-- Navigate to an internal page -->
<div data-role="content">
    <a href="#contact" data-role="button">Contact Us</a>
</div>
```

当一个单页面转换到另外一个单页面时，导航模型是不同的。例如，我们可以从多页面中提取出 contact 页面，然后命名为 contact.html 文件。现在我们的 home 页面（hijax.html）可以通过一个普通的 HTTP 链接引用来返回 contact 页面。

程序清单 2-4 Ajax 导航

```
<div data-role="content">
    <a href="contact.html" data-role="button">Contact Us</a>
</div>
```

When clicking the "Contact Us" link above, jQuery Mobile will process that request as follows:

当单击上面的“Contact Us”链接时，jQuery Mobile 将会按照如下步骤处理该请求。

1. jQuery Mobile 会解析 href，然后通过一个 Ajax 请求（Hijax）载入页面。为了获得一个直观了解，请看图 2-3。如果成功载入页面，则该页面会添加到当前页面的 DOM 中。

当页面成功添加到 DOM 中后，jQuery Mobile 可以根据需要来增强该页面，更新基础（base）元素的@href，并设置 data-url 属性（如果没有被显式设置的话）。

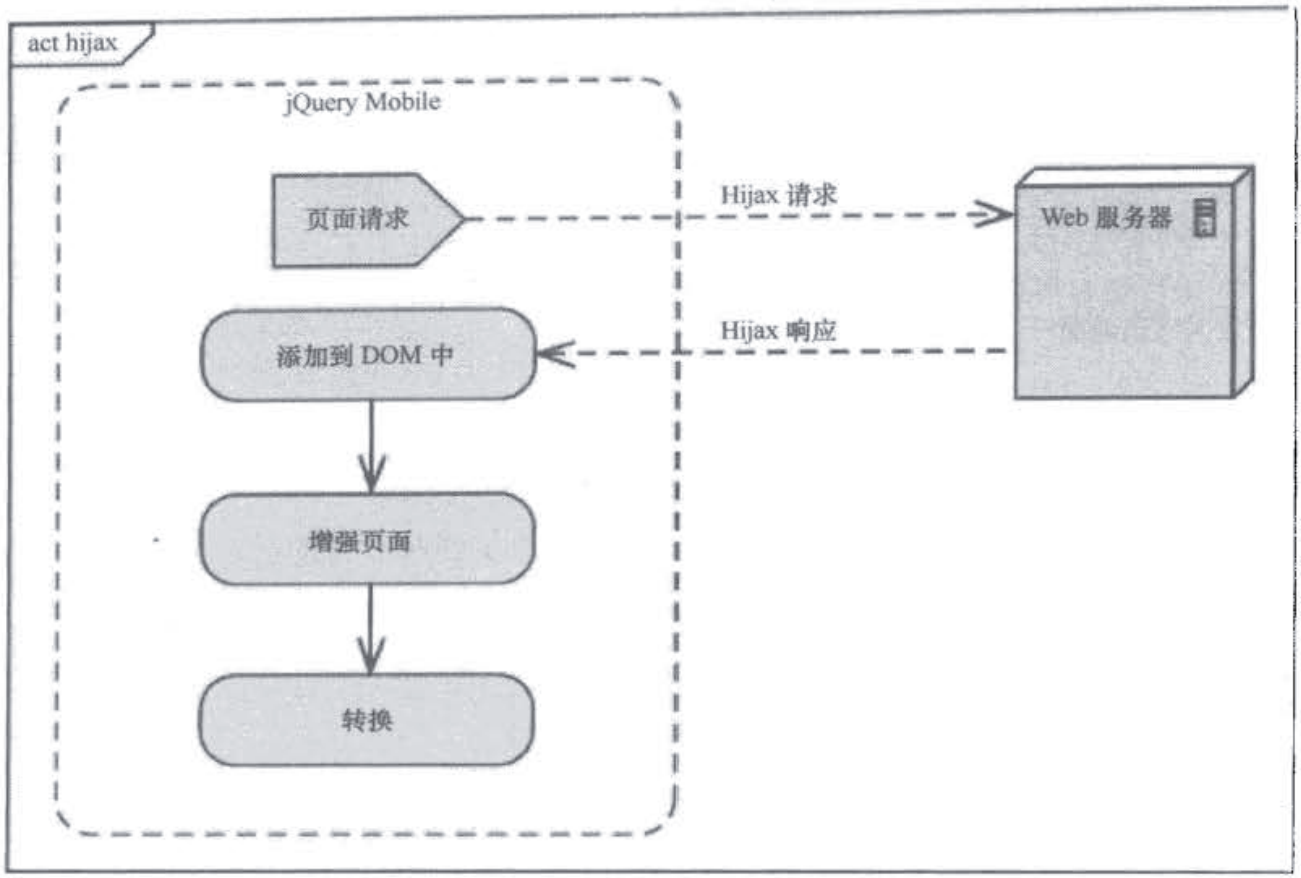


图 2-3 jQuery Mobile Hijax Request

2．框架随后使用应用的默认“滑动”转换模式转换到一个新的页面。框架也可以实现无缝的 CSS 转换，因为“from”页面和“to”页面都存在于 DOM 中。在转换完成之后，当前可见的页面或活动页面将会被指定为“ui-page-active”CSS 类。

3．产生的 URL 也可以作为书签。例如，如果想深链接（deep link）到 contact 页面，则可以通过其完整的路径来访问：

http://<host:port>/ch2/contact.html

注意：作为一个额外的优点，如果浏览器支持 HTML5 的 pushState，则基于 Ajax 的导航也可以在该浏览器中产生完全正确的 URL。桌面式 Safari、Chrome、Firefox 和 Opera 的最近版本都支持 HTML5 的这个特性。Android（2.2+）和 iOS5 也支持 pushState。在不支持该特性的浏览器中，基于#的 URL（http://<host:port>/hijax.html#contact.html）将会用来保持共享 URL 和对 URL 添加书签的能力。

4. 如果页面载入失败，则会显示和淡出一条短的错误消息，该消息是对“Error Loading Page（页面载入错误）”消息的覆写（overlay）（见图 2-4）。

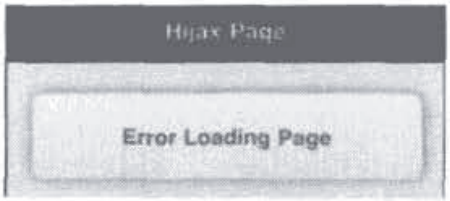


图 2-4 载入错误画面

2.3.1 $.mobile.changePage()

■ changePage( )函数处理从一个页面转换到另一个页面时涉及的所有细节。你可以转换到除当前页面之外的任何页面。可用的转换类型的完整列表如表 2-1 所示。

2.3.2 用途

■ $.mobile.changePage(toPage, [options])

2.3.3 参数

■ toPage(string 或 jQuery 集合)。将要转向的页面。

■ toPage(string)。一个文件 URL(“contact.html”)或内部元素的 ID(“#contact”)。

■ toPage(jQuery 集合)。包含一个页面元素的 jQuery 集合，而且该页面元素是该集合的第一个参数。

■ options(object)。配置 changePage 请求的一组键/值对。所有的设置都是可选的。

■ transition(string, default: $.mobile.defaultTransition)。为 changePage 应用的转换。默认的转换是“滑动”。

■ reverse(boolean, default:false)。指示该转换是向前转换还是向后转换。默认的转换是向前。

■ changeHash(boolean, default:ture)。当页面转换完成之后，更新页面 URL 的址

■ role(string，default:“page”)。在显示页面时使用的 data-role 值。如果页面是对话框，则使用“dialog”。

■ pageContainer(jQuery 集合，default:$.mobile.pageContainer)。指定应该包含载入页面的元素。

■ type (string, default:“get”)。在生成页面请求时，指定所使用的方法(get 或 post)。

■ data (string 或 object, default:undefined)。发送给一个 Ajax 页面请求的数据。

■ reloadPage (boolean, default: false)。强制页面重新载入，即使它已经位于页面容器的 DOM 中。

■ showLoadMsg (boolean, default: true)。在请求页面时，显示载入信息。

■ fromHashChange(boolean, default: false)。指示 changePage 是否来自于一个 hashchange 事件。

示例 1：

```
//Transition to the "contact.html" page.
$.mobile.changePage( "contact.html" );

<!-- Markup equivalent -->
<a href="contact.html">Contact Us</a>
```

示例 2：

```
// Go to an internal "#contact" page with a reverse "pop" transition.
$.mobile.changePage( '#contact', { transition: "pop", reverse: true } );

<!-- Markup equivalent -->
<a href="contact.html" data-transition="pop" data-direction="reverse">
  Contact
</a>
```

示例 3：

```
/* Dynamically create a new page and open it */

// Create page markup
var newPage = $("<div data-role=page data-url=hi><div data-role=header>
  <h1>Hi</h1></div><div data-role=content>Hello Again!</div></div>");

// Add page to page container
newPage.appendTo( $.mobile.pageContainer );

// Enhance and open new page
$.mobile.changePage( newPage );
```

重要： Ajax 导航不能用于载入一个外部链接的情况。

```
<!-- Ajax navigation will be ignored when loading a page with a
rel="external" or target attribute -->
<a href="multi-page.html" rel="external">Home</a>

<!-- Ajax navigation will be ignored -->
<a href="multi-page.html" target="_blank">Home</a>
```

在这样的情况下，将会发生正常的 HTTP 请求过程。而且，也无法应用 CSS 转换。前面提到，通过动态地将“from”和“to”页面载入到同一个 DOM，框架可以实现平滑的转换，而且随后也可以应用一个平滑的 CSS 转换。不使用 Ajax 导航，则无法实现平滑转换，而且在转换期间，也不会显示默认的载入消息（$.mobile.loadingMessage）。

2.3.4 配置 Ajax 导航

Ajax 导航是全局启用的，当你很在意 DOM 的大小时，或者是你需要支持的某个特定设备不支持 hash 历史更新时（见下面的“注意”），则可以禁用这个特性。默认情况下，jQuery Mobile 可以为我们管理 DOM 的大小或缓存，它只将活动页面转换所涉及的“from”和“to”页面合并到 DOM 中。要禁用 Ajax 导航，在绑定移动初始事件时，设置$.moible.ajaxEnabled=false。有关配置 jQuery Mobile 或管理 DOM 缓存的更多详情，请见第 8 章。

注意： 在所有已知与 hash 历史更新有冲突的平台上，Ajax 导航已经被禁用。例如，jQuery Mobile 已经为 BlackBerry 5、Opera Mini(5.0-6.0)、Nokia Symbian 3 和 Windows Phone 6.5 禁用了 Ajax 导航（$.mobile.ajaxEnabled=false）。在使用普通的 HTTP 和全页面刷新时，这些设备更为实用。

2.4 转换

在页面之间进行转换时，jQuery Mobile 有 6 个可供选择的基于 CSS 的转换效果。默认情况下，框架会为所有的转换应用“滑动”效果。通过为任意链接、按钮或表单添加 data-transition 属性，我们可以设置其他的转换效果。

```
<a href="dialog.html" data-transition="slideup">Show Dialog</a>
```

转换效果的完整列表见表 2-1。

表 2-1 转换效果

转换	常见的用途
滑动（slide）	在页面之间移动的最常见的转换。在一个页面流中，该转换给出了向前移动或向后移动的外观。这是所有链接之间的默认转换
卷起（slideup）	用于打开对话框或显示额外信息的一个常见的转换。该转换给出的外观可以用来为当前活动的页面收集额外的输入信息
向下滑动（slidedown）	该转换与卷起相对，但是可用于实现类似的效果
弹出（pop）	用于打开对话框或显示额外信息的另一个转换。该转换给出的外观可以用来为当前活动的页面收集额外的输入信息
淡入/淡出（fade）	用于入口页面或出口页面的一个常见的转换效果
翻转（flip）	用于显示额外信息的一个常用转换。通常情况下，屏幕的背景会显示没有必要存在于主 UI 上的配置选项（信息图标）
无（none）	不应用任何转换

页面到页面的转换过程按照如下步骤发生。

1. 用户轻敲按钮，以导航到下一个页面（见图 2-5）。

2. 框架使用一个 Hijax 请求载入下一个页面，然后添加到当前页面的 DOM 中。当前页面和下一个页面实际上是并排放置的，因此准备发生一个平滑转换（见图 2-6）。

3. 框架转换到下一个页面（见图 2-7）。该示例使用了默认的“滑动”转换。

4. 下一个页面得以显示，转换完成（见图 2-8）。

图 2-5 步骤 1：轻敲按钮，以导航到另外一个页面

图 2-6 步骤 2：框架载入并排放置的下一个页面

图 2-7 步骤 3：框架转换到下一个页面

图 2-8 步骤 4：转换完成

提示：通过为你的链接添加 data-direction="reverse"属性，可以设置一个“向后（backward）”转换。向前的“滑动”转换是向左滑动，因此向后的“滑动”转换是向右滑动。例如，当向历史访问页面转换时，默认情况下使用的是向后转换。但是，如果你的标题上有一个“home”链接，你需要应用 data-direction="reverse"属性，否则将会产生默认的“向前”转换效果。

```
<a href="home.html" data-icon="home" data-iconpos="notext"
   data-direction="reverse" class="ui-btn-right jqm-home">
   Home
</a>
```

2.5 对话框

对话框与页面相似，只不过对话框的边界是有间距的（inset），从而产生模态对话框（modal dialog）的外观。在对话框的设计方面，jQuery Mobile 相当灵活。我们可以创建确认对话框（见图 2-9）、警告对话框（见图 2-10），甚至是动作表单样式的对话框（见图 2-11 和图 2-12）。

图 2-9 确认对话框

图 2-10 警告对话框

我们可以将一个页面转换为链接或页面组件上的一个对话框。在一个链接中，添加 data-rel="dialog" 属性，如程序清单 2-5 所示。添加这个属性之后，将会自动载入目标页面，并将其增强为一个模态对话框。

程序清单 2-5 链接级别的转换

```
<!-- Open a page as a dialog -->
<a href="#terms" data-rel="dialog" data-transition="slidedown">Terms</a>


<!-- The page remains unchanged. -->
<div data-role="page" id="terms">
 <div data-role="header">
  <h1>Terms and Conditions</h1>
 </div>

 <div data-role="content">
  Do you agree to these terms?


  <a href="#" data-role="button" data-inline="true"
     data-rel="back" data-theme="a">Disagree</a>
  <a href="#" data-role="button" data-inline="true">Agree</a>
 </div>
</div>
```

我们也可以在页面容器上配置对话框。将 data-role="dialog"属性添加到页面容器中，当该页面容器组件在载入页面时，其将会被增强为一个模态对话框（见程序清单 2-6）。

程序清单 2-6 页面级别的转换

```
<!-- Link without data-rel="dialog" attribute -->
<a href="#terms" data-transition="slidedown">Terms and Conditions</a>


<!-- Configure this page to appear as a dialog -->
<div data-role="dialog" id="terms">
 <div data-role="header">
  <h1>Terms and Conditions</h1>
 </div>

 <div data-role="content" data-theme="c">
  Do you agree to these terms?

  <a href="#" data-role="button" data-inline="true"
     data-rel="back" data-theme="a">Disagree</a>
  <a href="#" data-role="button" data-inline="true">Agree</a>
 </div>
</div>
```

> **注意：**带有 data-rel="dialog"属性的任何链接，或者是带有 data-role="dialog"属性的任何页面不会出现在页面访问历史记录中，因此无法添加书签。例如，如果你导航到了一个对话框，关闭该对话框并轻敲浏览器的向前按钮，你无法进入到刚才的那个对话框，因为它不存在于历史记录中。

2.5.1 链接与页面配置的对比

既然有两个选项可用于打开对话框，我们该选择哪一个呢？我个人倾向于选择页面配置（data-role="dialog"），因为我们只需要在页面容器中配置一次对话框，而且导航到该对话框的按钮也无需任何修改。例如，如果有 3 个按钮链接到我们的对话框，基于页面的配置则只需要修改一次。而基于链接的配置则需要修改 3 次，每一次对应一个按钮。

jQuery Mobile 对话框 API 还公开了一个 close 方法，当需要以程序方式来处理对话框时，可以使用该方法。例如，如果我们想使用程序来处理图 2-9 中“Agreee”按钮的进程，我们可以处理单击事件，然后处理任何需要的业务逻辑，并在完成之后关闭对话框。

```
function processAgreement(){
  // Save the agreement...

  // Close the dialog
  $('.ui-dialog').dialog('close');
}
```

2.5.2 操作表

除了传统的对话框之外，我们还可以将对话框设计为一个操作表（action sheet），见图 2-11 和图 2-12。只需移除标题，添加较少的样式（styling）更新（见程序清单 2-7），你的对话框就成为一个操作表。操作表通常用来请求一个来自用户的响应。为了获得最佳的用户体验，建议为操作表使用“向下滑动”转换。为方便起见，当对话框关闭时，会自动应用相反的转换。例如，当你关闭某动作表单时，将会应用“卷起”转换。

图 2–11 操作表#1

图 2–12 操作表#2

程序清单 2-7 操作表(ch2/action-sheet1.html)

```
<!-- Logout link -->
<a href="#logout" data-transition="slidedown">Logout</a>


<!-- Create an action sheet by simply removing the header! -->
<div data-role="dialog" id="logout">
  <div data-role="content">
    <span class="title">Are you sure?</span>

    <a href="#home" data-role="button" data-theme="b">Yes, I'm Sure!</a>
    <a href="#" data-role="button" data-theme="c" data-rel="back">No Way!</a>
  </div>
 <style>
   span.title { display:block; text-align:center;
                margin-top:10px; margin-bottom:20px; }
 </style>
</div>
```

这也是我们研究的第一个 data-theme 属性。通过使用该属性，我们可以简单地为所有的 jQuery Mobile 组件添加对比度和样式。在对话框示例中，我们可以设置背景和按钮的主题。当设计对话框按钮时，通常会为取消按钮和动作按钮的样式添加对比度。jQuery Moible 内的主题将在第 7 章详细讨论。

2.5.3 对话框 UX 指南

在设计你的 UI 组件时，一致性是最重要的设计目标。就对话框相关的指南而言，来自苹果的移动界面指南[4]的一些提示如下所示。

> **提示：**默认情况下，对话框的最大宽度为 500 像素。在某些较小的移动设备上显示时，将会铺满整个屏幕；而在台式机或者平板电脑上显示时，则只占据 500 像素的宽度。如果你想要覆写这个默认的宽度，可以在你的主题中使用如下 CSS。
>
> ```
> ui-dialog .ui-header, .ui-dialog .ui-content, .ui-dialog .ui-footer {
> max-width: 100%; }
> ```

1. 警告

■ 使用警告来显示可以影响应用程序使用的重要信息（见图 2-10）。警告不是用户发起的。

■ 警告按钮要么是浅颜色，要么是深颜色。对于单按钮的警告来说，按钮总是浅颜色的。对于一个包含两个按钮的对话框，左边的按钮总是深颜色的，而右边的按钮总是浅颜色的（见图 2-9）。

■ 在一个包含两个按钮的对话框中，如果提出了一个肯定的动作，而且用户很有可能会选择这个动作，则取消该动作的按钮应该位于右边，而且是浅颜色的。通常情况下，执行有风险的动作的按钮是红色的。

2. 操作表

■ 使用操作表来收集用户发起的任务的确认信息（见图 2-11）。操作表也可以针对当前的任务为用户提供一系列选项（见图 2-12）。

■ 一个操作表至少包含两个按钮，它可以让用户选择如何完成他们的任务。

■ 包含一个取消按钮，以允许用户放弃任务。取消按钮位于操作表的底部，以促使用户在做出选择之前，阅读了所有的选项。取消按钮的颜色应该与背景的颜色相同。

---

[4] 见 http://developer.apple.com/library/ios/documentation/userexperience/conceptual/ mob

2.6 带有媒体查询的响应式布局

要使用 jQuery Mobile 来创建响应式设计，建议使用 CSS3 Media Queries[5]。例如，如果你打算为一个特定设备的朝向增强布局，你可以使用媒体查询来检测设备的朝向，然后根据需要应用你的 CSS 修改。

```
@media (orientation: portrait) {
  /* Apply portrait orientation enhancements here... */
}

@media (orientation: landscape) {
  /* apply landscape orientation enhancements here... */
}
```

在某些情况下，jQuery Mobile 将会为你创建响应式设计。下面将讲解 jQuery Mobile 的响应式设计如何良好地应用于竖屏（portrait）模式和横屏（landscape）模式中的表单字段。例如，在竖屏视图中（见图 2-13），标签位于表单字段的上面。而当将设备横屏放置时（见图 2-14），表单字段和标签并排显示。这种响应式设计可以基于设备可用的屏幕真实状态提供最实用的体验。jQuery Mobile 为用户提供了很多这样优秀的 UX（用户体验）原则。

图 2-13 响应式设计（竖屏模式）

图 2-14 响应式设计（横屏模式）

[5] 见 http://www.w3.org/TR/css3-mediaqueries/.

警告： 如果在 iOS 中启动图 2-14，并切换到横屏模式，则可能会注意到，在 Mobile Safari[6] 中存在 iOS 缩放问题。当元视口标记被设置为 content="width=device-width, initial-scale=1"，或者是设置为允许用户缩放的任何值时，当将设备调整为横屏放置时，将会引起页面以大于 1 的比例进行放大，由此导致的结果是，页面的右边将被裁切掉一部分，用户必须双击（有时可能需要操作多次）页面，才能让页面以合适的尺寸进行缩放并显示。

在 Mobile Safari 解决掉该问题之前，你可以使用多个选项来修复这个问题。

- 可以禁用缩放。由于禁用缩放时，会破坏可访问性，因此在进行该操作时务必谨慎行事。

```
<meta name="viewport" content="width=device-width,
      minimum-scale=1.0, maximum-scale=1.0">
```

- 在用户进行缩放时，可以动态调整元标记[7]。

在前面的例子中（见图 2-13），通过使用 min-max 宽度媒体特性，jQuery Mobile 能够应用响应式设计。例如，当浏览器支持的宽度大于 450 像素时，表单元素可以浮动在它们的标签旁边。CSS 支持文本输入的这种行为，如下所示。

```
label.ui-input-text {
    display: block;
}

@media all and (min-width: 450px){
  label.ui-input-text { display: inline-block; }
}
```

重要： Windows Phone 7（Internet Explorer 8 和更低的版本）不支持媒体查询。如果想要在不支持媒体查询的浏览器上支持响应式设计，建议使用 Respond.js[8]。Respond.js 为不支持媒体查询的浏览器提供了该功能。

你可以找到一组数量有限的特定 Webkit 的媒体扩展。例如，如果要在具有高分辨率的 retina（视网膜）显示屏的新 iOS 设备上应用 CSS 增强，你可以使用 webkit-min-devic

6 见 http://filamentgroup.com/examples/iosScaleBug/.
7 见 http://adactio.com/journal/4470/.
8 见 https://github.com/scottjehl/Respond.

e-piexel-ratio 媒体特性：

```
//Webkit-specific media query for the iOS high-resolution Retina display @media screen
and (-webkit-min-device-pixel-ratio: 2){
  // Apply retina display enhancements
}
```

作为对 iOS 用户的一个额外奖励，jQuery Mobile 包含了一全套针对 retina 显示屏优化过的图标，这些图标能够自动应用到带有高分辨率显示屏的任何 iOS 设备上。

注意： 如果你选择将媒体特定的样式隔离存放到单独的文件中，你可以使用 HTML <link>媒体属性来引用它们。如果用户比较在意文件的隔离存放，这种做法确实不错，但是从性能角度来看，该方法并不能让人满意，因为每一个独立的文件都需要一个额外的 HTTP 请求。

```
<link href="default.css" />
<link media="all and (min-width:450px)" href="widescreen.css" />
```

2.7 总结

在本章中，我们讲解了 jQuery Mobile 的基本知识，以及如何迅速上手并运行一个 jQuery Mobile 应用程序。我们还讲解了 jQuery Mobile 的两种页面模板，并从性能和导航流方面讨论了每一种模板的各自优势。我们还看到了 jQuery Mobile 如何将语义标记增强为一个优化过的移动体验。此外，我们讨论了所有可用的页面转换，以及每种页面转换常见的使用模式。最后，我们讨论了设计对话框的多种方式，从而可以为通知用户或从用户那里收集回馈信息创建一个有效的界面。在第 3 章，我们将会进一步讲解如何使用 jQuery Mobile 来导航，以及如何充分使用页眉和页脚控件来管理移动应用程序的数据。

第3章

使用页眉、工具栏和标签栏来导航

所有的移动应用程序都需要工具栏来辅助屏幕中的导航或者管理数据。在本章，我们将介绍提供这些特性的 jQuery Mobile 组件。主要的组件是页眉和页脚。页眉通常用于显示页面标题，还可以包含控件，以辅助用户在屏幕中进行导航或管理对象。页脚与页眉相似，但是它的职责通常是使用工具栏或标签栏来管理的。此外，我们还要了解分段控件的功能。分段控件是一个可以放置在页眉或页脚的专用控件，它用来显示数据的其他视图。我们会讲解每一种组件，并演示如何使用文本、标准图标和自定义图标来设计它们。

3.1 页眉栏

页眉栏显示当前屏幕的标题。此外，也可以在上面添加用于导航的按钮，或者是添加用来管理页面中的项目的控件。尽管页眉是可选的，但是它通常用来提供活动页面的标题。我们先来看一下页眉的结构，然后再讨论如何为页眉添加额外的控件，以辅助管理页面中的项目。

3.1.1 页眉基础知识

与页眉相关的一些要点如下所示。

- 页眉使用 data-role="header"属性来定义。

■ 页眉是一个可选的组件。

■ 回退按钮不会在页眉中显示，除非你显式地启用了它。回退按钮将在下面的小节详细介绍。

■ 可以使用 date-theme 属性来调整页眉的主题。如果没有为页眉设置主题，则它会继承页面组件的主题。默认的主题是黑色的（black）（data-theme="a"）。

■ 在默认情况下，所有的页眉级别（H1～H6）具有相同的风格，以维持视觉上的连贯性。

■ 通过添加 data-position="fixed"属性，可以对页眉进行固定。

> **提示：** 你也可以使用页眉作为一个分段控件，如图 3-5 所示。分段控件允许用户显示相关数据的不同视图。

3.1.2 页眉结构

页眉的一个基本用途是仅显示活动页面的标题。页眉最简单的形式如下所示。

```
<div data-role="header">
  <h1>Header Title</h1>
</div>
```

3.1.3 页眉定位

有 3 种样式可以用于定位页眉，如下所示。

■ Default（默认）：默认的页眉会在屏幕的顶部边缘显示，而且在屏幕滚动时，页眉将会滑到可视范围之外。

```
<div data-role="header">
  <h1>Default Header</h1>
</div>
```

■ Fixed（固定）：固定的页眉总是位于屏幕的顶部边缘位置，而且总是保持可见。但是，在屏幕滚动的过程期间，页眉是不可见的，当滚动结束之后，页眉才出现。通过添加 data-position="fixed"属性，可以创建一个固定的页眉。

```
<div data-role="header" data-position="fixed">
  <h1>Fixed Header</h1>
</div>
```

> **注意**：为了实现真正固定的工具栏，浏览器需要支持 fixed 或 overflow:auto 这两种定位方式中的一种。幸运的是，WebKit 的新版本（iOS5）开始支持这一行为。在 jQuery Mobile 中，通过将 touchOverflowEnabled 配置选项设置为 true，可以启用该行为（更多细节，请见第 8 章的“配置 jQuery Mobile 选项”一节）。

■ Responsive（响应式）：当我们创建一个全屏页面时，页面中的内容会全屏显示，而页眉和页脚则基于触摸响应来出现或消失。对显示照片和播放视频来说，全屏模式相当有用。要创建一个全屏的页面，在页面容器中添加 data-fullscreen=“true”，然后在页眉和页脚元素中添加 data-position=“fixed”属性（见图 3-1）。例如，在图 3-1 中，我们有一个用来显示照片的全屏页面。如果用户轻敲屏幕，则页眉和页脚将会出现和消失（见图 3-2）。在这个例子中，我们有一个照片查看器，而且其页眉显示照片的计数信息，页脚显示一个工具栏以辅助导航、发送电子邮件或删除照片。

程序清单 3-1 全屏（ch3/position-fullscreen.html）

```
<div data-role="page" data-fullscreen="true">
  <div data-role="header" data-position="fixed">
    <h3>Header</h3>
  </div>

  <div data-role="content">
    <!-- Fullscreen content -->
  </div>

  <div data-role="footer" data-position="fixed">
    <h3>Footer</h3>
  </div>
```

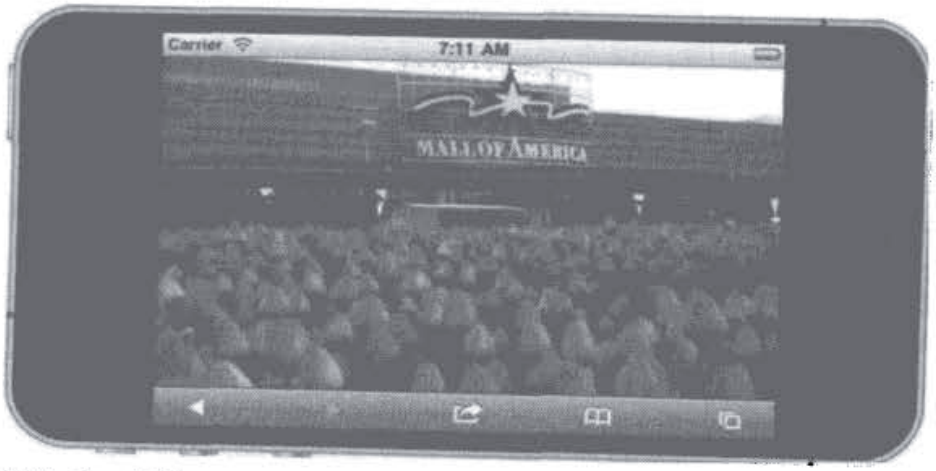


图 3-1 全屏

图 3-2 具有响应式页眉和页脚的全屏

> **注意**：在查看 jQuery Mobile 页面时，在 iOS 和 Android 系统中浏览器的 URL 地址栏会隐藏起来。这个特性很方便，可以让用户查看更多的屏幕尺寸，而且可以实现平滑转换。但是，如果你需要查看 URL 地址栏，向下拖动页面，URL 地址栏则会变得可见。

3.1.4 页眉按钮

有些情况下，你可能需要在页眉中添加控件，以辅助管理屏幕内容。例如，在编辑数据时，保存和取消按钮是经常会用到的两个控件。可以添加到页眉中的按钮有 3 种类型，如下所示。

- 只带有文本的按钮。
- 只带有图标的按钮（见图 3-4）。只带有图标的按钮需要添加两个属性：data-icon 和 data-iconpos="notext"。data-icon 值的完整列表，请见表 4-1。
- 既有文本又有图标的按钮（见图 3-3）。这种类型的按钮也需要 data-icon 属性。

每一种类型的按钮示例如下所示。

```
<!-- A button with only text -->
<a href="#">Done</a>

<!-- A button with only an icon -->
<a href="#" data-icon="plus" data-iconpos="notext"></a>

<!-- A button with text and an icon -->
<a href="#" data-icon="check">Done</a>
```

3.1.5 既有文本又有图标的按钮

在图 3-3 中，页眉中带有一个“Cancel”按钮和一个“Done”按钮，用来辅助管理电影评论的条目。在程序清单 3-2 中，按钮被设计为一个普通的链接。我们可以通过 data-icon 属性为每一个按钮附加一个图标。在页眉内部，按钮依据它们的语义顺序进行摆放。例如，第一个按钮是左对齐的，第二个按钮是右对齐的。如果页眉只包含一个按钮，你可以通过将 class=“ui-btn-right”属性添加到按钮的标记中，来右对齐按钮。

程序清单 3-2 页眉按钮

```
<div data-role="header" data-position="inline">
  <a href="#" data-icon="delete">Cancel</a>
  <h1>Add Review</h1>
  <a href="#" data-icon="check">Done</a>
</div>
```

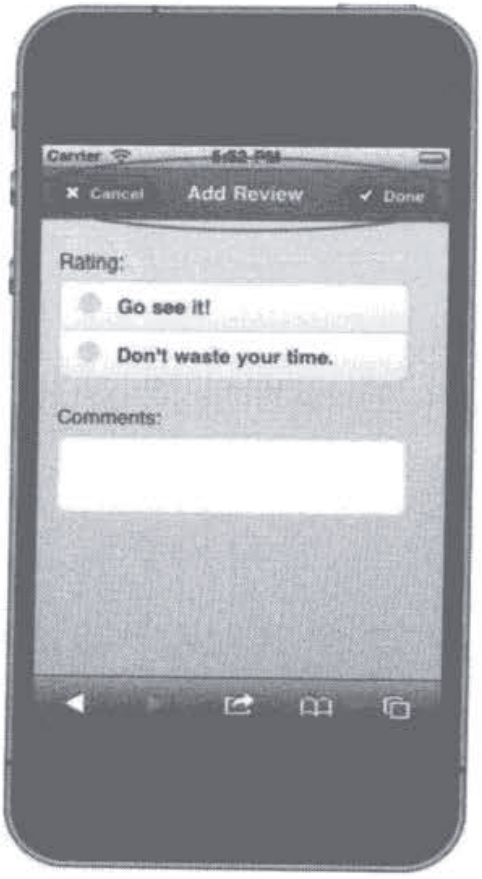


图 3-3 带有按钮的页眉

3.1.6 只带有图标的按钮

jQuery Mobile 包含多个标准图标（见表 4-1），你可以用它们来创建只带有图标的

按钮。例如，“info”图标通常与“翻转（flip）”转换一起使用，来显示配置选项或更多的信息。标准图标在使用时，只占用很小的屏幕空间，而且它们的含义在所有的设备上都是相对一致的。例如，如果我们想要添加一个条目到现有的一个列表中，我们可以选择一个“plus”图标，用户通过该图标可以添加一个条目到列表中（见图 3-4）。在本例中，我们有一个电影评论列表，用户可以轻敲“add”图标来创建他们的评论。为了创建一个只带有图标的按钮，需要添加两个属性，如程序清单 3-3 所示。

程序清单 3-3 带有图标的页眉

```
<div data-role="header">
  <h1>Reviews</h1>
  <a href="#" data-icon="plus" data-iconpos="notext" class="ui-btn-right"></a>
</div>
```

图 3-4 带有图标的页眉

3.1.7 带有分段控件的页眉栏

分段控件是一组内联（inline）的控件，其中每一个控件可以显示一个不同的视图

例如，图 3-5 中的分段控件可以按照特定的分类来显示电影。该分段控件允许用户通过他们选择的分类（In Theatres、Coming Soon 或 Top Rated）来迅速查看电影。

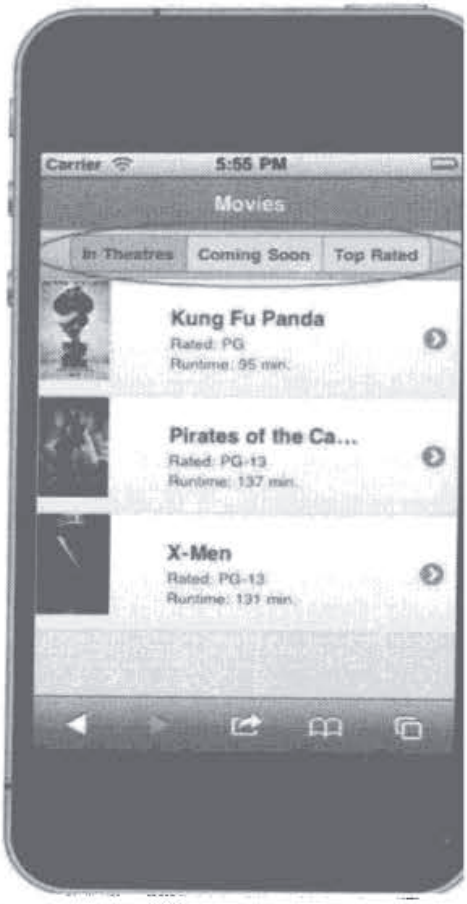


图 3-5 分段控件

建议将分段控件放置在主页眉内，如程序清单 3-4 所示。如果将页眉作为一个固定控件来放置，则这种放置方式可以让分段控件与主页眉无缝集成。通过添加少量的样式更新，我们可以拥有一个允许用户以不同视图来快速查看数据的分段控件。

程序清单 3-4 分段控件（ch3/header-segmented-control.html）

```
<div data-role="header" data-theme="b" data-position="fixed">
  <h1>Movies</h1>
  <div class="segmented-control ui-bar-d">
    <div data-role="controlgroup" data-type="horizontal">
      <a href="#" data-role="button" class="ui-control-active">
        In Theatres
      </a>
      <a href="#" data-role="button" class="ui-control-inactive">
        Coming Soon
      </a>
      <a href="#" data-role="button" class="ui-control-inactive">
        Top Rated
```

```
    </div>
  </div>
</div>


<style>
  .segmented-control { text-align:center;}
  .segmented-control .ui-controlgroup { margin: 0.2em; }
  .ui-control-active, .ui-control-inactive {
      border-style: solid; border-color: gray; }
  .ui-control-active { background: #BBB; }
  .ui-control-inactive { background: #DDD; }
</style>
```

3.1.8 修复被截断的页眉或页脚

如果页眉或页脚的标题过长，则 jQuery Mobile 会进行截断处理（见图 3-6）。当文本太长时，jQuery Mobile 会截断文本，并在文本的末尾添加一个省略号。如果你遇到了这种情况，并且希望显示完整的文本（见图 3-7），则可以调整 CSS 选择器，来修复该问题，如程序清单 3-5 所示。

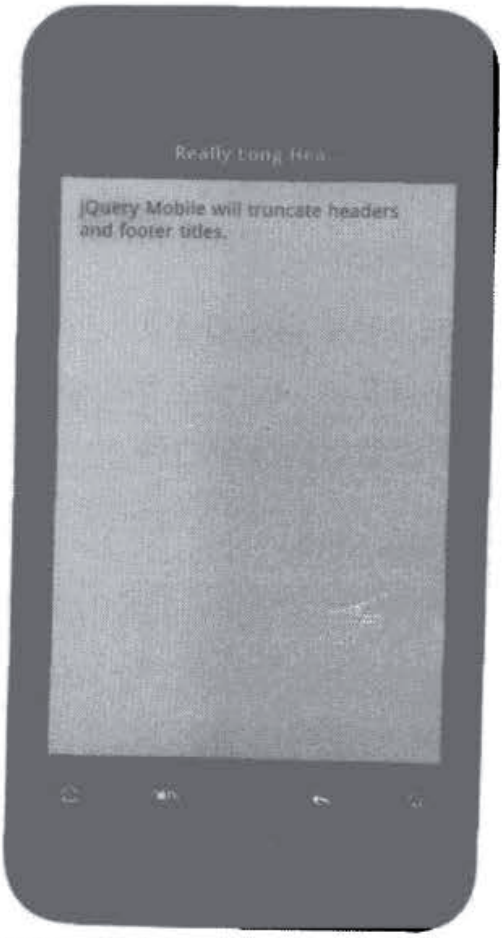


图 3-6 截断问题

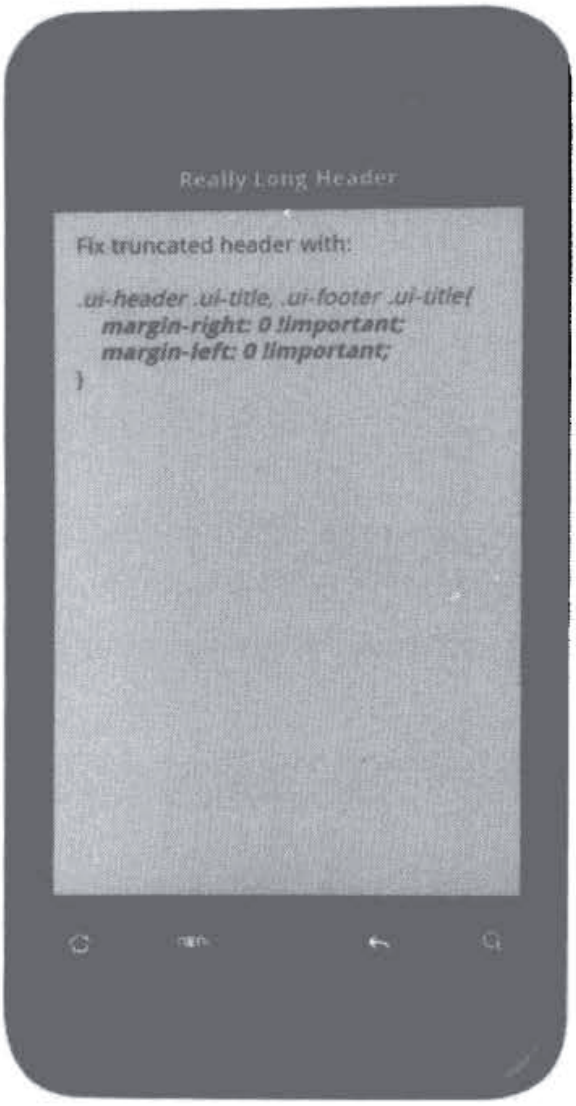


图 3-7 截断修复

程序清单 3-5 截断问题的修复（ch3/truncation-fixed.html）

```
.ui-header .ui-title, .ui-footer .ui-title {
  margin-right: 0 !important; margin-left: 0 !important;
}
```

3.2 回退按钮

回退（Back）按钮在 UX 设计人员中引发了巨大的争论。在某些设备和所有的浏览器中，我们是应该添加自己的回退按钮呢，还是应该使用硬件/软件自带的回退按钮呢？幸运的是，jQuery Mobile 能够在全局自动启用或禁用该按钮。我们还可以逐个页面添加或移除该按钮。

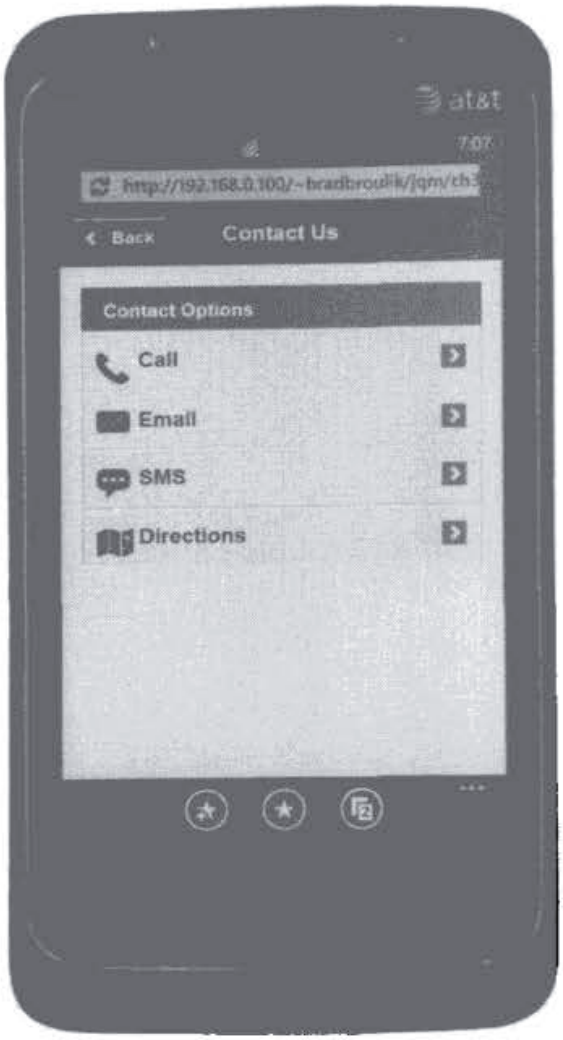


图 3-8 回退按钮必须被显式启用

在 jQuery Mobile 中，回退按钮在默认情况下是禁用的。如果想要让回退按钮出现在页眉内，可以有多个选择来进行添加。

■ 通过在页面容器中添加 data-auto-back-btn="true"属性，可以为某个特定页面添加回退按钮。

■ 在绑定 mobileinit 选项时，通过将 addBackBtn 选项设置为 true，可以在全局启用回退按钮。在设置该选项之后，如果有页面存在于历史访问记录中，回退按钮会自动显现。在后台，回退按钮只是执行 window.history.back()方法。如下面的代码所示，我们也可以重写回退按钮的默认文本和主题。例如，通常会使用上一个页面的标题来标记（label）回退按钮。data-back-btn-text 属性可以用于该惯例。有关设置全局配置选项的更多详情，请见第 8 章。

```
<!-- Show the back button and override the default back button text -->
<div data-role="page" data-add-back-btn="true"
     data-back-btn-text="Previous">

// Globally enable the back button, set the default back button text,
// and set back button theme
$(document).bind('mobileinit',function(){
    $.mobile.page.prototype.options.addBackBtn = true;
    $.mobile.page.prototype.options.backBtnText = "Previous";
    $.mobile.page.prototype.options.backBtnTheme = "b";
});
```

此外，如果在全局启用了回退按钮，可以通过在页面页眉中添加 data-add-back-btn="false"属性，禁用特定页面上的回退按钮。这会将回退按钮从特定页面的页眉中移除。

```
<!-- Disable the back button on a specific page if we globally enabled it -->
<div data-role="header" data-add-back-btn="false">
```

提示：尽管回退按钮可以用于所有的移动浏览器，但是在 jQuery Mobile 的某些特定情况下，我们可能会明确地需要使用回退按钮或其他导航。

■ 建议所有的页面都包含一个返回主屏幕的链接，这可以通过一个链接图标（logo）或主页按钮来实现。这样做的目的是，防止用户在导航时不会进入死胡同。当用户访问一个深链接或采用书签标记过的页面时，可能经常会出现这样的情况。如果你唯一的导航机制是回退按钮，而且历史访问记录是空的，则自动回退按钮不会显现，这样用户就进入了死胡同。因此，一个非常好的解决方法是在页眉栏的右侧包含一个主页图标链接。

■ 在为 PhoneGap 集成进行设计时，如果你的目标 OS（比如 iOS 或 WebOS）不支持基于硬件的导航，则需要考虑使用回退按钮。

3.2.1 回退链接

如果希望创建一个行为与回退按钮相类似的按钮，则可以为任何锚元素添加 data-rel="back"属性。

```
<a href="home.html" data-rel="back" data-role="button">Go Back</a>
```

通过利用 data-rel="back"属性，链接将会模拟回退按钮，返回一个历史条目（window.history.back()），并忽略链接的默认 href 值。对于 C 级浏览器或不支持 JavaScript 的浏

览器来说，它们会忽略 data-rel，而且将 href 属性作为一个备用。

3.3 页脚栏

除了细微的差别之外，页脚组件和页眉组件几近相同。两者之间主要的区别在按钮的放置方面，页脚要更加灵活。例如，在使用页眉时，第一个按钮是左对齐的，而第二个按钮是右对齐的。而页脚则是以从左到右的顺序直线放置它的按钮。这个灵活性允许我们将页脚设计为工具栏或标签栏。在查看这两种类型的案例之前，先来了解一下页脚的基本知识。

3.3.1 页脚基础知识

与页脚相关的一些要点如下所示。

- 页脚使用 data-role="footer"属性来定义。
- 页脚按照从左到右的顺序直线放置它的按钮。这种灵活性可以用来创建工具栏或标签栏。
- 页脚是一个可选的组件。
- 使用 data-theme 属性可以调整页脚的主题。如果不为页脚设置主题，则它会继承页面组件的主题。默认的主题是黑色的（data-theme="a"）。
- 通过添加 data-position="fixed"属性，可以固定页脚的位置。
- 在默认情况下，所有的页脚级别（H1～H6）具有相同的风格，以维持视觉上的一致性。

3.3.2 页脚结构

最简单的页脚形式如下面的代码所示。data-role="footer"是唯一需要的属性。在页脚内，可以包含任何语义 HTML。页脚通常包含工具栏和标签控件。工具栏提供了一组用户可以在当前环境中使用的动作。标签栏则可以允许用户在应用程序内的不同视

图之间进行切换。

```
<div data-role="footer">
  <!-- Add footer text or buttons here -->
</div>
```

> **提示：** 为了将页脚定位在屏幕的最底部，可以为页脚元素添加 data-position="fixed"。在默认情况下，页脚位于内容的后面，而不是屏幕的底部边缘（见图 3-9）。例如，如果你的内容只占据了一半的屏幕高度，则页脚将会出现在屏幕的中央位置。通过为页脚元素添加 data-position="fixed" 属性，我们可以将页脚固定在屏幕的底部。
>
> ```
> 
> ```

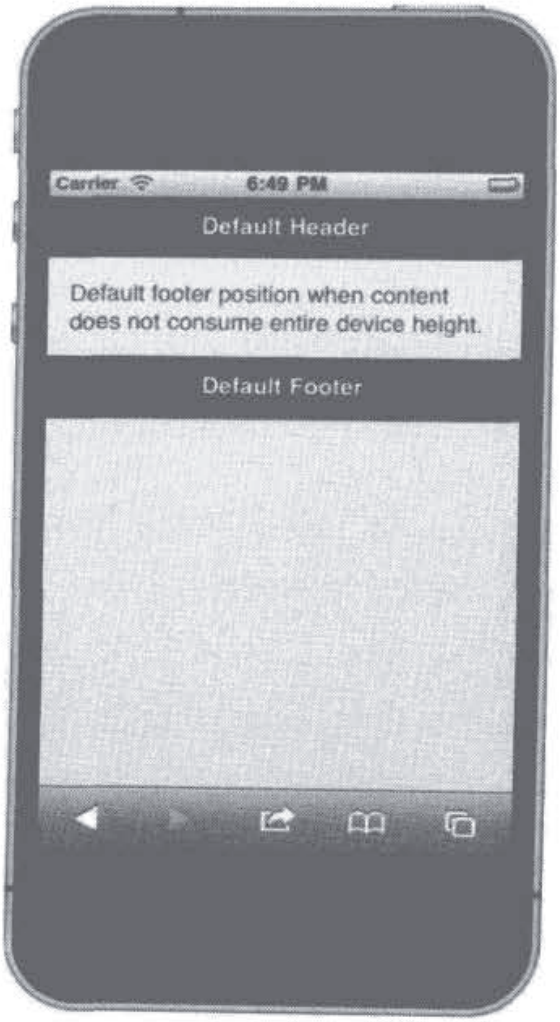


图 3-9 默认的页脚位置

3.3.3 页脚定位

用于定位页眉的 3 种样式同样也适用于页脚，如下所示。

■ Default（默认）：默认的页脚会在内容区域的后面显示。例如，如果你的内容超出了视口的高度，则只有在屏幕滚动到内容的最底部时，才能看到页脚。

```
<div data-role="footer">
  <!-- Default footer -->
</div>
```

■ Fixed（固定）：固定的页脚总是位于屏幕的底部边缘位置，而且总是保持可见。但是，在用户滚动屏幕的过程中页脚是不可见的，当滚动结束之后页脚才出现。通过添加 data-position="fixed"属性，可以创建一个固定的页脚。

```
<div data-role="footer" data-position="fixed">
  <h3>Fixed Footer</h3>
</div>
```

■ Responsive（响应式）：当我们创建一个全屏页面时，页面中的内容会出现在整个屏幕，而页眉和页脚则基于触摸响应来出现或消失。对显示照片和播放视频来说，全屏模式相当有用。要创建一个全屏的页面，在页面容器中添加 data-fullscreen="true"，然后在页眉和页脚元素中添加 data-position="fixed"属性。相关示例请见图 3-1。

3.3.4 页脚按钮

可以添加到页脚中的 3 种按钮样式如下所示。

■ 只带有文本的按钮。这种样式的按钮可以用在工具栏内，原因是工具栏的外观没有标签栏那么大。页脚内一个正常的链接会作为一个只带有文本的按钮来显示。

```
<a href="#">Sync</a>
```

■ 只带有图标的按钮。这种样式的按钮也可以用于工具栏中。只带有图标的按钮需要添加两个属性：data-icon 和 data-iconpos="notext"。

```
<a href="#" data-icon="plus" data-iconpos="notext"></a>
```

■ data-icon 值的完整列表，请见表 4-1。

■ 既有文本又有图标的按钮。这种样式的按钮可以用于标签栏内。

```
<a href="#" data-icon="home">Home</a>
```

提示： jQuery Mobile 是一个相当出色的框架，可以用来构建在移动设备、平台电脑和台式机浏览器中显示的应用程序。在移动设备上，页眉和页脚提供了一种“本地”的感觉，而在台式机上进行查看时，页眉和页脚的转换性能较差。如果你的 jQuery Mobile

> 应用程序是针对不同的浏览器大小而设计的，你可能倾向于删除页眉和页脚组件。作为一种替代方法，通过直接在内容区域内添加自定义的页眉和页脚标记，无疑会更有利。

3.4 工具栏

工具栏可用来辅助管理当前屏幕中的内容。例如，邮件应用程序通常使用工具栏来管理电子邮件。当用户需要执行与当前屏幕中的对象相关联的动作时，工具栏会非常有用。在构建工具栏时，我们可以选择使用图标或文本。在下面的工具栏示例中，它包含了图标按钮、文本按钮以及一个分段控件。

3.4.1 带有图标的工具栏

只有图标构成的工具栏相当常见。它的主要优势是，与文本构成的工具栏相比，它占据的屏幕空间更少。选择图标时要选择能够表达正确含义的标准图标，这一点很重要。在图 3-10 中，我们有一个显示电影评论的屏幕。为了帮助用户管理评论，我们可以利用一个由标准图标构成的工具栏。工具栏允许用户执行 5 种动作：

图 3-10 带有标准图标的工具栏

1. 导航到前面的评论。

2. 答复评论。

3. 将评论标记为最喜欢的评论。

4. 添加一条新的电影评论。

5. 导航到后面的评论。

创建工具栏时，仅需要最少的标记（见程序清单 3-6）。在带有 data-role="navbar" 属性的 div 中，我们只需要其中包含按钮的一个无序列表即可。工具栏按钮相当灵活，而且可以根据设备的宽度进行等间距排放。在本例中，我们使用的按钮来自于 jQuery Mobile 的标准按钮套件（见表 4-1）。

程序清单 3-6 工具栏(ch3/toolbar-icons-standard.html)

```
<div data-role="footer" data-position="fixed">
  <div data-role="navbar">
    <ul>
      <li><a href="#" data-icon="arrow-l"></a></li>
      <li><a href="#" data-icon="back"></a></li>
      <li><a href="#" data-icon="star"></a></li>
      <li><a href="#" data-icon="plus"></a></li>
      <li><a href="#" data-icon="arrow-r"></a></li>
    </ul>
  </div>
</div>
```

提示：如果你想为导航栏（navbar）添加一些特殊的效果，而自定义了一些图标，则这些自定义图标同样也适用于导航栏组件。如果对自定义图标解决方案感兴趣，我们将会在程序清单 3-10 中为你展示。

3.4.2 带有分段控件的工具栏

可以在工具栏中放置一个分段控件，从而让用户通过不同的视角来访问应用程序的数据，或者为用户提供一个不同的应用程序视图。在图 3-11 中，我们在工具栏内放置了分段控件，以允许用户显示他们的日历数据的不同视图。你可能已经注意到，这一分段控件与我们前面的页眉案例中使用的分段控件相同。这意味着，我们可以同时在页眉和页脚组件中使用分段控件。分段控件只是一组包含在一个控件组内并按照你

的要求进行风格化的按钮。在下一节，我们将讲解分段控件在标签栏中的用途。

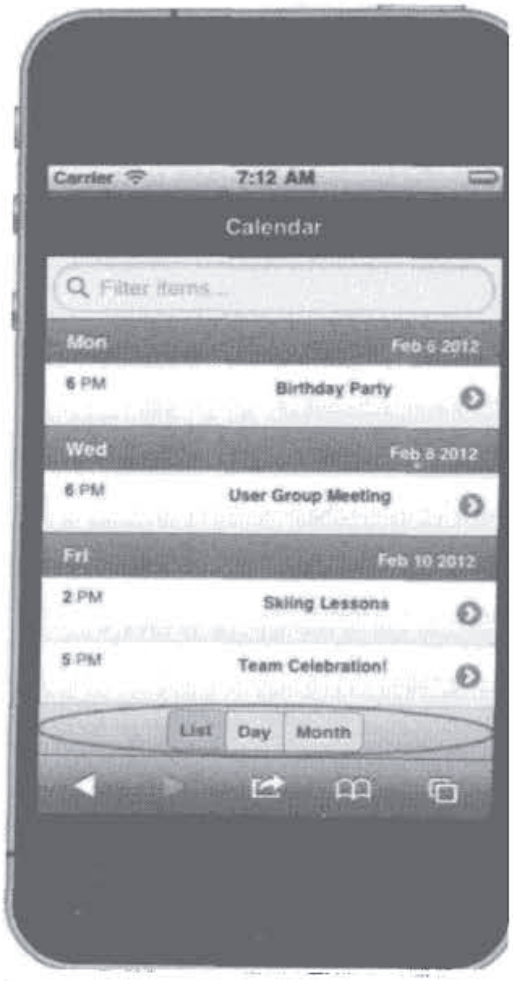


图 3-11 带有分段控件的工具栏

程序清单 3-7 带有分段控件的工具栏(ch3/toolbar-segmented-control.html)

```
<!-- Toolbar with a segmented control -->
<div data-role="footer" data-position="fixed" data-theme="d"
     class="segmented-control">
  <div data-role="controlgroup" data-type="horizontal">
    <a href="#" data-role="button" class="ui-control-active">List</a>
    <a href="#" data-role="button" class="ui-control-inactive">Day</a>
    <a href="#" data-role="button" class="ui-control-inactive">Month</a>
  </div>
</div>

<style>
  .segmented-control { text-align:center; }
  .segmented-control .ui-controlgroup { margin: 0.2em; }
  .ui-control-active, .ui-control-inactive { border-style: solid;
        border-color: gray; }
  .ui-control-active { background: #BBB; }
  .ui-control-inactive { background: #DDD; }
</style>
```

3.5 标签栏

我们也可以将页脚设计为一个标签栏。通过标签栏，用户可以以不同的视图来查看应用程序。其实，标签栏的行为与 Web 上可以见到的基于标签的导航相类似。标签栏通常作为一个永久的页脚出现在屏幕的底部边缘，而且用户可以在应用程序的任何位置访问它。出于清晰性考虑，标签栏通常包含同时显示图标和文本的按钮。在下面的例子中，我们将遇到 3 种样式的标签栏。其中，第一个标签栏示例包括 jQuery Mobile 内可用的标准图标。第二个标签栏示例使用的是自定义图标。而在最后一个标签栏示例中，我们会将标签栏与同一个 UI 内的分段控件结合起来，从而允许用户通过同一个屏幕导航，以不同形式查看数据。

3.5.1 带有标准图标的标签栏

最简单的标签栏解决方案（见图 3-12）使用的是 jQuery Mobile 的标准图标集，如程序清单 3-8 所示。jQuery Mobile 标准图标的完整列表如表 4-1 所示。如果你使用这些标准图标，则标签栏无需任何额外的样式风格。

图 3-12 带有标准图标的标签栏

程序清单 3-8 带有标准图标的标签栏(ch3/tabbar-icons-standard.html)

```
<!-- tab bar with standard icons -->
<div data-role="footer" data-position="fixed">
  <div data-role="navbar">
   <ul>
    <li><a href="#" data-icon="home">Home</a></li>
    <li><a href="#" data-icon="star" class="ui-btn-active">
           Movies</a></li>
    <li><a href="#" data-icon="grid">Theatres</a></li>
   </ul>
  </div>
</div>
```

3.5.2 永久标签栏

为了让标签栏永久显现，我们需要为页脚添加一个额外的属性。为了在页面转换期间，页脚可以一直显现，可以为每一个标签栏的页脚添加 data-id 属性，并将其值设置为相同的识别符。例如，在程序清单 3-9 中，每一个标签栏包含一个 data-id="main-tabbar"的识别符。通过添加该属性，标签栏则可以在页面转换期间一直显现。例如，如果我们轻敲一个不活动的标签栏，而且在页面切换期间屏幕将会“滑动”，而标签栏仍然保持为固定和永久显现的状态。此外，在从一个标签转换到另外一个标签时，为了保持每一个标签栏的活动状态，需要添加 ui-state-persist 和 ui-btn-active 类。用于永久标签栏的标记在下面的程序清单中以黑体显示。

程序清单 3-9 永久标签栏

```
<!-- Movies tab bar -->
<div data-role="footer" class="tabbar" data-id="main-tabbar"
     data-position="fixed">
  <div data-role="navbar" class="tabbar">
    <ul>
      <li><a href="tabbar-movies.html"
             class="ui-btn-active ui-state-persist">Movies</a></li>
      <li><a href="tabbar-theatres.html">Theatres</a></li>
   </ul>
  </div>
</div>

<!-- Theatres tab bar -->
<div data-role="footer" class="tabbar" data-id="main-tabbar"
     data-position="fixed">
  <div data-role="navbar" class="tabbar">
    <ul>
      <li><a href="tabbar-movies.html">Movies</a></li>
      <li><a href="tabbar-theatres.html"
             class="ui-btn-active ui-state-persist">Theatres</a></li>
    </ul>
  </div>
</div>
```

3.5.3 带有自定义图标的标签栏

想在你的标签栏或工具栏中添加自定义图标吗？jQuery Mobile 通过添加最少的必要标记来提供对自定义图标的支持。例如，在下面的标签栏示例中（见图 3-13），包含了几个来自 Glyphish[1]的第三方图标。

图 3-13 带有在自定义图标的标签栏

为了支持自定义图标的添加，我们需要添加 data-icon="custom"属性，以及用于定位的一些自定义样式和 id，其中 id 用于将每个按钮与它的样式进行关联。这些新添加的内容在程序清单 3-10 中以黑体显示。

[1] 见 http://glyphish.com/。这是由 Joseph Wain 创建并且符合知识共享书名许可协议 3.0 版美国许可证（Creative Commons Attribution 3.0 United States License）许可的图标。

> **提示：**自定义按钮解决方案也同样适用于工具栏。事实上，只需将文本从按钮中移除，也就创建了一个带有自定义图标的瘦身型（slim）工具栏。

程序清单 3-10 带有自定义图标的标签栏(ch3/tabbar-icons-custom.html)

```
<!-- tab bar with custom icons -->
<div data-role="footer" class="ui-navbar-custom" data-position="fixed">
  <div data-role="navbar" class="ui-navbar-custom">
   <ul>
    <li><a href="#" id="home" data-icon="custom">Home</a></li>
    <li><a href="#" id="movies" data-icon="custom"
           class="ui-btn-active">Movies</a></li>
    <li><a href="#" id="theatres" data-icon="custom">Theatres</a></li>
   </ul>
  </div>
</div>


<style>
  .ui-navbar-custom .ui-btn .ui-btn-inner {
     font-size: 11px!important;
     padding-top: 24px!important;
     padding-bottom: 0px!important;
  }
  .ui-navbar-custom .ui-btn .ui-icon {
     width: 30px!important;
     height: 20px!important;
     margin-left: -15px!important;
     box-shadow: none!important;
     -moz-box-shadow: none!important;
     -webkit-box-shadow: none!important;
     -webkit-border-radius: none !important;
     border-radius: none !important;
  }
  #home .ui-icon {
     background: url(../images/53-house-w.png) 50% 50% no-repeat;
     background-size: 22px 20px;
  }
  #movies .ui-icon {
     background: url(../images/107-widescreen-w.png) 50% 50% no-repeat;
     background-size: 25px 17px;
  }
  #theatres .ui-icon {
     background: url(../images/15-tags-w.png) 50% 50% no-repeat;
     background-size: 20px 20px;
  }
</style>
```

3.5.4 带有分段控件的标签栏

到此为止，我们已经查看了标签栏和分段控件的示例，如何将

我们可以使用永久标签栏来辅助站点导航，而且可以使用分段控件来显示数据的不同视图。在下面的例子中（见图 3-14），我们创建了一个 UI，该 UI 允许用户在 Home、Movies 和 Theatres 标签之间进行导航。当用户选择 Moives 标签时，我们将在页眉内显示分段控件，以允许用户筛选他们的电影列表。在本例中，我们已经彻底移除了页眉文本，原因是活动标签已经用来突出显示页面的标题。有关该示例的完整源代码，请见 ch3/tabbar-and-segmented-control.html。

图 3–14 带有分段控件的标签栏

3.6 总结

在本章中，我们几乎讲解了 jQuery Mobile 内每一个页眉和页脚可能构成的组合。jQuery Mobile 有一套丰富的组件，它们可以大大简化导航和数据管理的需求。我们讲

解了用来在应用程序内提供不同视图的标签栏解决方案，以及用来管理当前屏幕中的对象的几个工具栏配置。我们还讲解了如何添加分段控件，以方便用户访问应用程序数据的不同视图。此外，在外观方面，每一个组件都相当灵活，而且每一个组件都是可主题化的，我们可以用图标、文本抑或是两者的组合来设计按钮。在下一章，我们会讲解所有可能的按钮样式（styling）选项，以及我们在 jQuery Mobile 内进行基于表单的开发时，可能会用到的组件。

第4章

表单元素和按钮

移动应用程序必须支持高效的用户体验。出于这个原因，人们很少看到带有众多表单字段的移动应用程序。事实上，app 与用户的互动越少，app 和用户将会越高效。移动 Web 正在逐步采用设备 API[1]，以允许开发人员通过最低限度的用户交互，收集丰富的信息。例如，74%的移动开发人员在他们的 app 内都使用了地理定位（geolocation）[2]。地理定位允许我们通过轻敲一个确认按钮，来收集用户所在的国家、州、城市、邮编，以及地址信息。尽管这些设备 API 可以让用户体验更为高效，但是对于设备不支持地理定位的用户来说，我们仍然需要使用传统的表格字段方式来捕获数据。

在本章中，我们首先讲解最流行的移动 UI 组件：按钮。按钮可以用多种方式来设计和配置。我们将会看到带有文本、图标，以及同时使用了这两者的按钮示例。

接下来，我们将详细讲解每一个标准的 HTML 表单组件，以及它们所适用的常见使用案例（use case）。用户很到很惊奇的一点是，jQuery Mobile 能够自动优化每一个表单组件，它的这个特性方便地提供了一个跨所有设备的统一用户体验。我们还会讲解 jQuery Mobile 的数据属性，这些属性对每一个表单元素来说都是唯一的。本章还会涵盖一些代码示例，在这些示例中，我们可以修改前面提到的这些数据属性，以配置和设计我们的表单。此外，我们还会讲解与每个表单组件相关的插件，并掌握当用户需要更为动态的体验时，如何使用插件 API 来动态创建、增强和更新我们的组件。

---

1 见 http://www.w3.org/2009/dap.

2 见 http://www.webdirections.org/sotmw2011/.

最后，我们还会研究 Mobisscroll 插件的特性，该特性为日期选择器、搜索过滤器和自定义列表提供了一个得体而且灵活的接口。

4.1 按钮

按钮是移动 app 内最常使用的控件，原因是它们可以提供一个非常高效的用户体验。在前面的许多例子中（其中包括对话框、动作表单、分段控件和页眉），我们已经使用到了按钮。jQuery Mobile 有多种形式的按钮，有链接按钮、表单按钮、图像按钮、只带有图标的按钮，以及同时带有文本和图标的按钮。正如所料，jQuery Mobile 按钮都具有一致的样式风格。无论你使用链接按钮还是基于表单的按钮，jQuery Mobile 框架都会以完全相同的方式对待它们。在讲解这些按钮时，我们也会获悉每一种按钮的常见使用案例。

4.1.1 链接按钮

链接按钮是最常使用的按钮类型。当需要将一个普通链接设计为按钮时，需要为链接添加 data-role="button"属性（见图 4-1）。

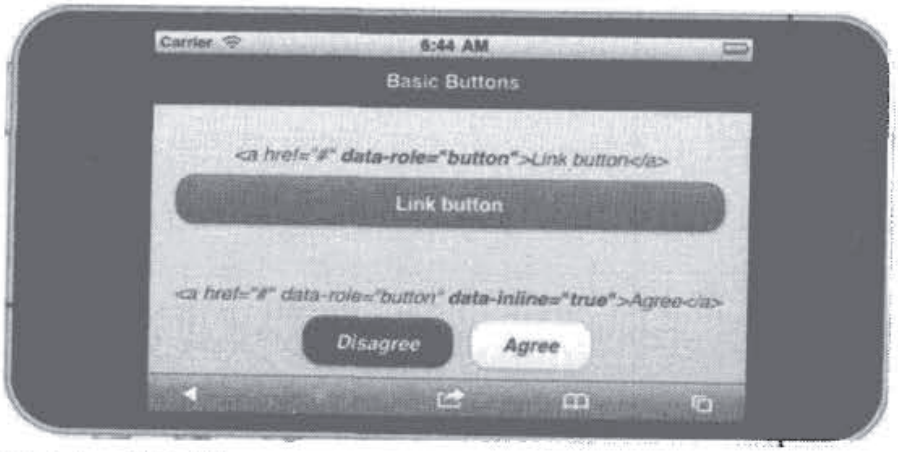


图 4-1 链接按钮

默认情况下，在页面的内容区域内的按钮都被设计为块级元素，这样可以填充其外层容器（即内容区域）的整个宽度。但是，如果需要的是一个更为紧凑的按钮，使其宽度与按钮内部的文本和图标的宽度相同，则可以添加 data-inline="true"属性（见程序清单 4-1）。

程序清单 4-1 链接按钮（ch4/link-buttons.html）

```
<a href="#" data-role="button">Link button</a>
<a href="#" data-role="button" data-inline="true">Disagree</a>
<a href="#" data-role="button" data-inline="true">Agree</a>
```

> **注意：**如果希望让按钮并排放置，并占据屏幕的整个宽度，则可以使用一个两列的网格。第 6 章会详细讲解灵活的网格布局。至于两列网格的具体布局，请参考程序清单 6-2。

4.1.2 表单按钮

基于表单的按钮（见程序清单 4-2）实际上要比基于链接的按钮更容易设计，这是因为我们无需进行任何修改。简单起见，框架会自动为用户将任何 button 或 input 元素转换为移动类型的按钮（见图 4-2）。

程序清单 4-2 表单按钮（ch4/form-buttons.html）

```
<button type="submit">Button element</button>
<input type="button" value="button" />
<input type="submit" value="submit" />
<input type="reset" value="reset" />
```

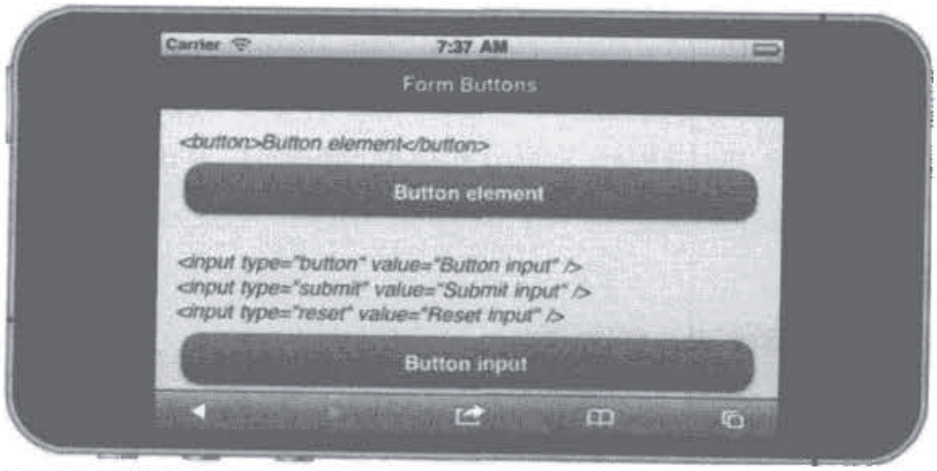


图 4-2 表单按钮

> **提示：**如果想要禁用表单按钮或任何其他控件的自动初始化，可以为这些元素添加 data-role="none"属性，这样，jQuery Mobile 就不会增强这些控件：
>
> ```
> <button data-role="none">Button element</button>
> ```

4.1.3 图像按钮

我们几乎不需要付出，就可以将图像设计为按钮。当使用锚标记来包含图像时，无需做任何修改（见图 4-3，其相关代码见程序清单 4-3）。但是，在将图片附加到一个 input 元素时，需要添加 data-role="none"属性。

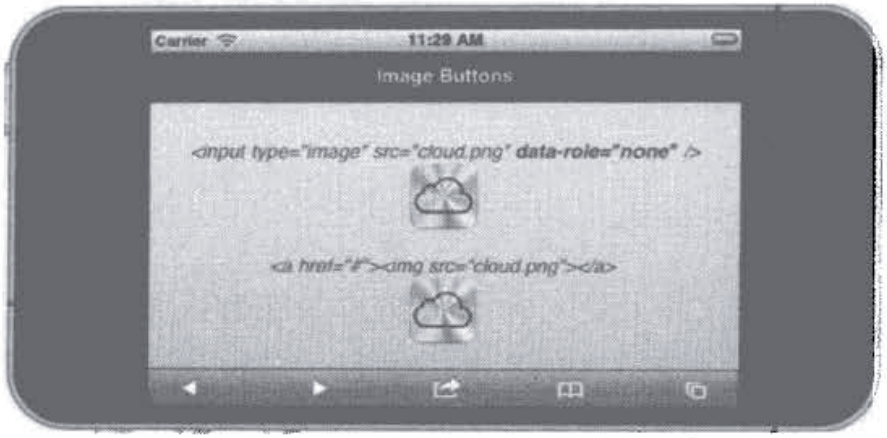


图 4-3 图像按钮

程序清单 4-3 图像按钮（ch4/image-buttons.html）

```
<!-- Image buttons -->
<input type="image" src="cloud.png" data-role="none" />
<a href="#"><img src="cloud.png"></a>
```

4.1.4 使用图标来设计按钮

jQuery Mobile 包含一组经常在移动应用程序中使用的标准图标，其中包含一个单独的白色图标精灵（sprite），而且该图标后面还有一个半透明的黑圈，以确保图标能够与任何背景色区分开来（见图 4-4）。

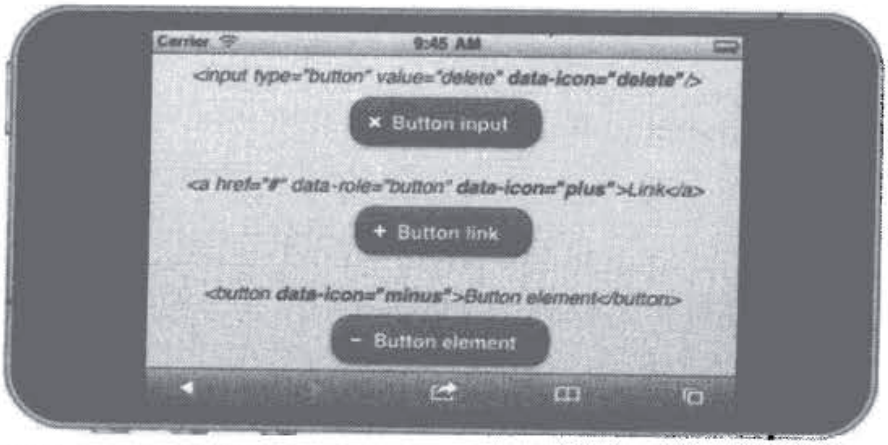


图 4-4 带有标准图标的按钮

通过添加 data-icon 属性并指定要显示的图标，可以将图标添加到任何按钮（见程序清单 4-4）。

程序清单 4-4 带有图标的按钮（ch4/icon-buttons-standard.html）

```
<!-- Buttons with standard icons.  Refer to Table 4-1 for icon list. -->
<input type="button" value="Delete" data-icon="delete"/>
<a href="#" data-role="button" data-icon="plus">Button link</a>
<button data-icon="minus">Button element</button>
```

表 4-1 包含了所有的 data-icon 属性值，以及与之相对应的图标图像。除了 data-icon="custom"属性之外，每一个属性值都有相对应的图像。在下一节，我们会讲解使用自定义图标的示例。

表 4-1 data-icon 列表

data-icon	Image
plus	
minus	
delete	
arrow-r	
arrow-l	
arrow-u	
arrow-d	
check	
gear	
refresh	
forward	
back	
grid	
star	
alert	
info	
home	
search	
custom	

4.1.5 只带有图标的按钮

由于只带有图标的按钮占据相当小的屏幕，因此通常用于页眉、工具栏和标签栏内（见图 4-5）。

图 4-5 只带有图标的按钮

在上一章，我们遇到了几个只带有图标的按钮示例。在图 3-4 中，我们首先遇到了一个“plus”图标，它允许用户轻敲“add”图标，以创建一个新的电影评论。我们还看到了在工具栏（见图 3-10）和标签栏（见图 3-12）内使用的只带有图标的按钮，它们可以用来表达每一个按钮的含义。要创建一个只带有图标的按钮，可以为该按钮添加 data-iconpos=“notext”属性（见程序清单 4-5）。

程序清单 4-5 只带有图标的按钮（ch4/icon-only-buttons.html）

```
<a href="" data-role="button" data-icon="plus" data-iconpos="notext"></a>
<button data-icon="search" data-iconpos="notext">Search</button>
```

注意：每一个白色图标后面的半透明黑圈可以确保与任何背景色形成对比，而且可以适用于 jQuery Mobile 主题系统。例如，在下面的图像中，第一排中的图标是使用 data-theme=“a”来设计（styled）的，而第二排使用的则是 data-theme=“c”。为了维持视觉上的一致性，建议创建 18×18 像素的白色图标，并保存为具有 Alpha 透明度的 PNG-8 图像类型。

4.1.6 按钮定位

默认情况下，图标是左对齐的（见图 4-6）。但是，通过为按钮添加 data-iconpos 属性，并指明需要对齐的位置，可以显式地将图标对齐在任何一侧（见程序清单 4-6）。

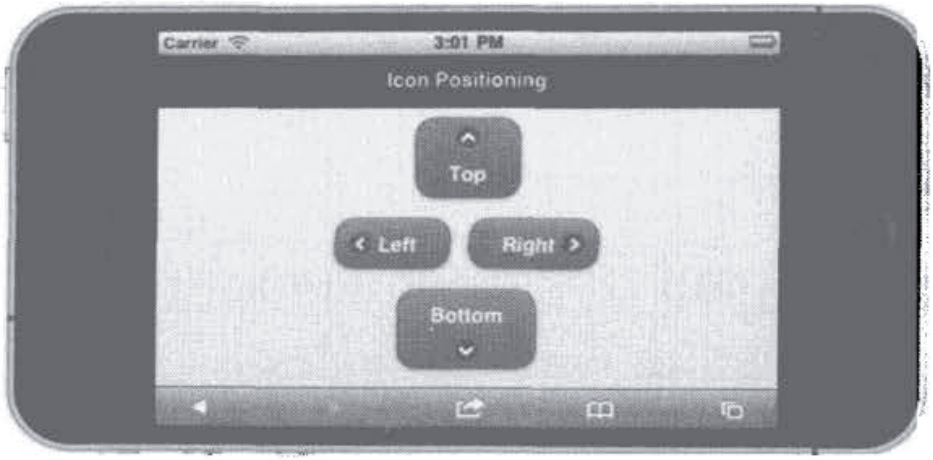


图 4-6 图标定位

程序清单 4-6 只带有图标的按钮（ch4/icon-positioning.html）

```
<a href="#" data-role="button" data-icon="arrow-u" data-iconpos="top">
<a href="#" data-role="button" data-icon="arrow-l" data-iconpos="left">
<a href="#" data-role="button" data-icon="arrow-r" data-iconpos="right">
<a href="#" data-role="button" data-icon="arrow-d" data-iconpos="bottom">
```

4.1.7 带有自定义图标的按钮

你应该还记得，在图 3-13 中，我们在标签栏中添加了自定义的 Glyphish 图标。我们可以用同样的方式，将自定义图标应用到按钮中（见图 4-7）。

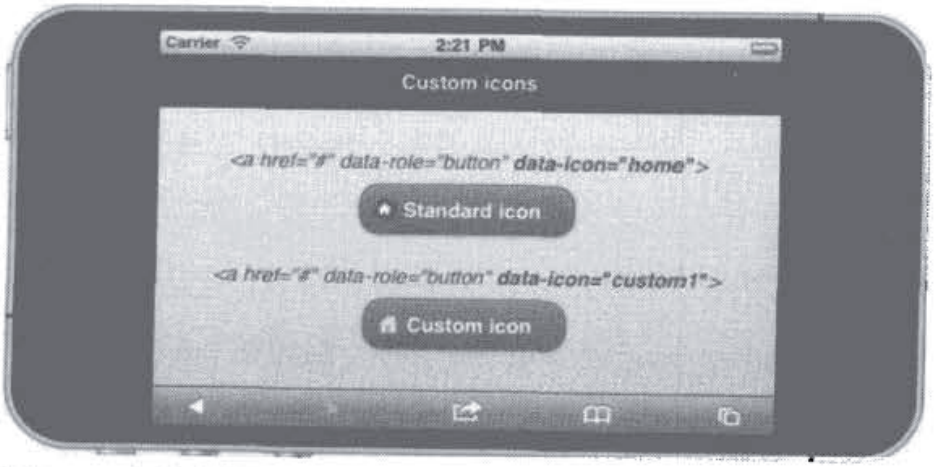


图 4-7 自定义图标

但是，我们可以为按钮应用更为简化的解决方案，如程序清单 4-7 所示。要为按钮添加自定义图标，需要采取两个步骤。

1．为链接添加 data-icon 属性。该属性的值必须唯一地标识自定义图标。例如，data-cion="my-custom-icon"。

2．创建一个 CSS 类属性，用于设置自定义图像的背景源。该类属性的名字必须被命名为".ui-icon-<data-icon-value>"。例如，如果 data-icon 值是"my-custom-icon"，则新创建的 CSS 类属性应该是".ui-icon-my-custom-icon"。

程序清单 4-7 在按钮中使用自定义图标（ch4/icon-buttons-lustom.html）

```
<style>
  .ui-icon-custom1 {
        background:url(...)50% 50% no-repeat;
        background-size: 14px 14px;
  }
</style>
<a href="#" data-role="button" data-icon="custom1">Custom</a>
```

> **提示：**用于自定义图像的背景源是使用数据 URI 方案（scheme）载入的。在从外部载入小图像时，这将是一个高性能的方法。例如，通过在线内（in-line）包含自定义图像，我们就不再需要 HTTP 请求。但是该技术的主要缺陷是，图像以 Base64 编码并形成字符串后，其尺寸要比原始的图像大 1/3。要查看完整的 Base64 编码的字符串，请见 ch4/icon-buttons/custom.html 中的源代码清单。

4.1.8 分组按钮

目前为止，在我们看到的按钮示例中，每一个按钮都是与其他按钮隔离的。但是，如果想对按钮进行分组，可以将按钮包含在一个控件组内。例如，第 3 章中的分段控件示例就是以这种方式分组的（见图 4-8）。

要实现这一效果，可以使用 data-role="controlgroup"属性将一组按钮包装在容器中（见程序清单 4-8）。

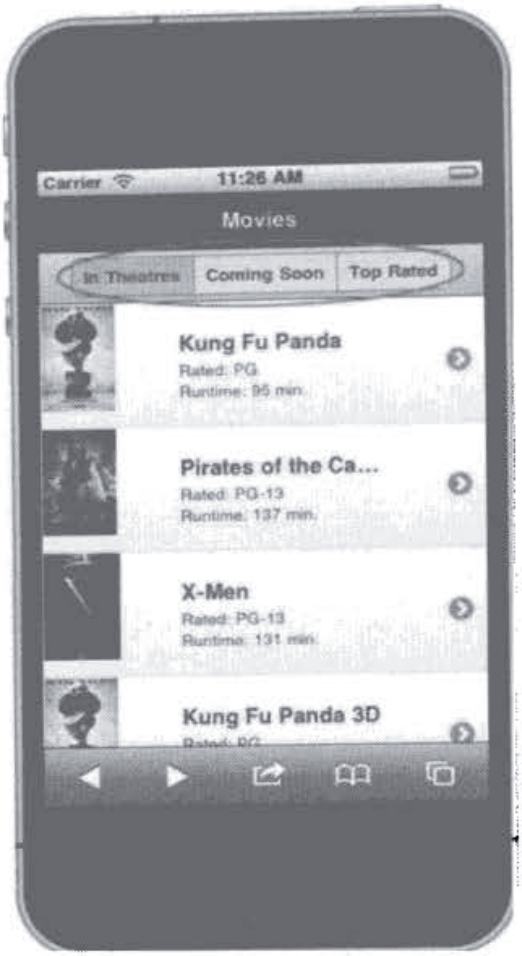


图 4–8 分组按钮

程序清单 4-8 分组按钮（ch3/header-segmented-control.html）

```
<div data-role="controlgroup" data-type="horizontal">
    <a href="#" data-role="button">In Theatres</a>
    <a href="#" data-role="button">Coming Soon</a>
    <a href="#" data-role="button">Top Rated</a>
</div>
```

默认情况下，框架会对按钮进行垂直分组，并移除所有的页边空白（margin），以及在按钮之间添加边界。此外，为了在视觉上增强分组，第一个和最后一个元素会使用圆角进行设计。

由于按钮在默认情况下是垂直摆放的，我们可以添加 data-type="horizontal"属性，从而水平摆放按钮。垂直摆放的按钮会占据其外层容器的整个宽度，而水平摆放的按钮的宽度则只与其内容一样宽。

> **注意：**在对按钮进行水平分组时，当控件组的宽度超出了屏幕宽度时，则会发生重叠现象。

4.1.9 主题按钮

按钮与所有的 jQuery Mobile 组件一样，都会继承其父容器的主题。此外，当需要使用不同颜色来设计按钮时，通过添加 data-theme 属性，可以为按钮应用任何所选择的任何主题（见程序清单 4-9）。

程序清单 4-9 主题按钮（ch2/action-sheet2.html）

```
<a href="#home" data-role="button" data-theme="b">YouTube</a>
<a href="#home" data-role="button" data-theme="b">Facebook</a>
<a href="#home" data-role="button" data-theme="b">Email</a>
<a href="#home" data-role="button" data-theme="c">Cancel</a>
```

例如，在对话框和动作表单示例中，为了提升按钮的可用性，我们按照第 2 章中的“对话框 UX 设计指南”设计了按钮（见图 4-9）。

图 4-9 主题按钮

4.1.10 动态按钮

button 插件（plugin）是一个能自动增强本地按钮的微件（widget）。我们可以使用该插件动态创建、启用和禁用按钮。如果需要在代码中动态创建按钮，可以有两个选择：通过标记驱动的方法动态创建按钮；显式设置 button 插件的选项。

在标记驱动的方法中，我们为新按钮创建 jQuery Mobile 标记，然后将其添加到内容容器中，然后再进行增强（见程序清单 4-10）。

程序清单 4-10 使用标记驱动的方法来创建动态按钮（ch4/dynamil-buttons.html）

```
// Add link button to content container and enhance it
$( '<a href="#" data-role="button" data-icon="star" id="b1">Star</a>' )
        .appendTo( ".ui-content" )
        .button();

// Add form button after the first button and enhance it
$( '<input type="submit" id="b2" value="Button 2" data-theme="a" />' )
        .insertAfter( "#b1" )
        .button();
```

对于插件驱动的方法而言，我们需要创建一个本地链接，将按钮插入到页面中，然后应用按钮增强（见程序清单 4-11）。

程序清单 4-11 使用插件驱动的方法来创建动态按钮（ch4/dynamie-buttons.html）

```
// Create a new button, insert it after button 2, and enhance it.
$( '<a href="#">Home</a>' )
        .insertAfter( "#b2" )
        .button({
                'icon':'home',
                'inline': true,
                'shadow': true,
                'theme': 'b'
        });
```

在最后一个例子中，我们创建多个表单按钮，但是我们不再为每个按钮分别调用 button 插件，而是通过一次触发页面容器的 create 方法，对所有的按钮进行增强（见程序清单 4-12）。我们也可以使用 button 插件的 enable 和 disable 方法，来动态启用或禁用按钮，如程序清单 4-12 所示。

程序清单 4-12 创建按钮，并动态禁用/启动它们（ch4/dynamic-buttons.html）

```
// Create multiple form buttons
$( '<button id="button3">Button3</button>' ).insertAfter( "#button2" );
$( '<button id="button4">Button4</button>' ).insertAfter( "#b
```

```
// Enhance all widgets on the page
$.mobile.pageContainer.trigger( "create" );

// Disable form button
$( "#button3" ).button( "disable" );

// Enable form button
$( "#button3" ).button( "enable" );
```

> 提示：触发页面容器的 create 方法会增强页面中的所有组件：$.mobile. page Container.trigger("create"); 当需要同时增强多个页面组件时，可以使用这种简便的方法。

1. 按钮选项

框架为了动态增强按钮而使用的 button 插件具有如下选项：

corners *boolean*

default: true

默认情况下，按钮是圆角的，将该选项设置为 false 则会移除按钮的圆角。该选项还可以公开作为一个数据属性：data-corners="false"。

```
$( "#button1" ).button({ corners: false });
```

icon *string*

default: null

设置按钮的图标。该选项还可以公开作为一个数据属性：data-icon="plus"。

```
$( "#button1" ).button({ icon: "home" });
```

iconpos *string*

default: "left"

设置图标的位置。可能的值有："left"、"right"、"top"、"bottom"和"notext"。"notext"值会将按钮显示为一个只带有图标而没有文本的按钮。该选项还可以公开作为一个数据属性：data-iconpos="notext"。

```
$( "#button1" ).button({ iconpos: "notext" });
```

iconshadow *boolean*

default: true

当该选项值为 true 时，框架会为图标添加阴影。该选项还可以公开作为一个数据属性：data-iconshadow="false"。

```
$( "#button1" ).button({ iconshadow: false });
```

initSelector *CSS selector string*

default: "button, [type='button'], [type='submit'], [type='reset'], [type='image']"

initSelectior 用来定义用于触发 widget 插件自动初始化的选择器(元素类型

数据角色[data role]等）。例如，由默认选择器匹配的所有元素都会由 button 插件进行增强。为了重写该选择器，可以绑定到 mobileinit 事件，并根据情况更新选择器。

```
$( document ).bind( "mobileinit", function(){
$.mobile.button.prototype.options.initSelector = "...";
});
```

inline *boolean*
default: false

如果该选项设置为 true，则按钮将以内嵌（inline）按钮的形式显示。默认情况下，按钮会占据其所在容器的整个宽度。相较而言，内嵌按钮只占据其文本的宽度。该选项还可以公开作为一个数据属性：data-inline="true"。

```
$( "#button1" ).button({ inline: true });
```

shadow *boolean*
default: true

默认情况下，按钮都有阴影。将该选项设置为 false，则会移除阴影。该选项还可以公开作为一个数据属性：data-shadow="false"。

```
$( "#button1" ).button({ shadow: false });
```

2. 按钮方法

button 插件具有如下方法。

enable: enable a disabled button

```
$( "#button1" ).button( "enable" );
```

disable: disable a button

```
$( "#button1" ).button( "disable" );
```

3. 按钮事件

button 插件支持如下事件。

create triggered when a button is created

在创建一个自定义按钮时，将触发该事件。它并不是用来创建一个自定义按钮。

```
$( '<a href="#" id="button2">Button2</a>' )
        .insertAfter( "#button1" )
        .button({
                theme: 'a',
                create: function(event) {
                        console.log( "Creating button..." );
                }
        })
```

4.2 表单元素

所有的本地表单元素在经由 jQuery Mobile 增强之后，能够更具吸引力，并更好地应用于移动设备。但是，不支持这些增强的老式浏览器会转而求助本地元素，以维护一个可用的体验。

4.2.1 表单基础知识

jQuery Mobile 用来构建基于表单的应用程序所采用的方法，与我们传统使用的构建 Web 表单的方法非常相似。尽管为了清晰起见，应该指明 action 和 method 属性，但是这并不是必需的。默认情况下，action 属性会默认为当前页面的相对路径，该路径可以通过$.mobile.path.get()找到，而未指定的 method 属性默认为“get”。

在提交表单时，通过默认的“滑动”转换，当前页面将会转换到后续页面。但是，通过我们之前用来管理链接的属性，我们可以配置表单转换行为（见程序清单 4-13）。

程序清单 4-13 提交表单（ch4/form-request.html）

```
<form action="/save.html" method="post" data-transition="pop">
        <label for="email">Email:</label>
        <input type="email" name="email" id="email" value="" />
        <button type="submit" name="submit">Submit</button>
</form>
```

我们可以在表单元素中添加如下属性，以管理转换或禁用 Ajax。

- data-transition="pop"
- data-direction="reverse"
- data-ajax="false"

> **警告：** 在整个站点中，要确保每一个表单的 id 属性都是唯一的，这很重要。前面提到，在转换时，jQuery Mobile 会同时将“from”页面和“to”页面载入到 DOM 中，以完成平滑的转换。为了避免任何冲突，表单 id 必须唯一。

> **提示：** 在构建表单时，建议使用一致的标签与每一个表单字段进行语义关联。label 的 for 属性和 input 的 id 属性将建立这种关系：
>
> ```
> <label for="name">Name:</label>
> <input type="text" name="name" id="name" value="" />
> ```

> 该关联将创建 508 兼容的应用程序，这些应用程序可以通过辅助技术来访问。而政府或国家机构经常要求的就是应用程序的可访问性。通过使用 WAVE[3]工具，可以测试你的应用程序的兼容性。

4.2.2 文本输入

文本输入是移动设备上最麻烦的表单字段。除非你是世界文本输入冠军，否则在物理或真实的 QWERTY 键盘上输入文字时，效率会非常低。这就是尽可能自动收集用户信息的价值所在。前面提到，设备 API 有助于简化这一用户体验。尽管最大限度地减少这些繁琐的任务是我们所期望的目标，但是有些时候，我们必须使用文本输入来收集用户的反馈信息。最常见的文本表单字段如图 4-10 所示。

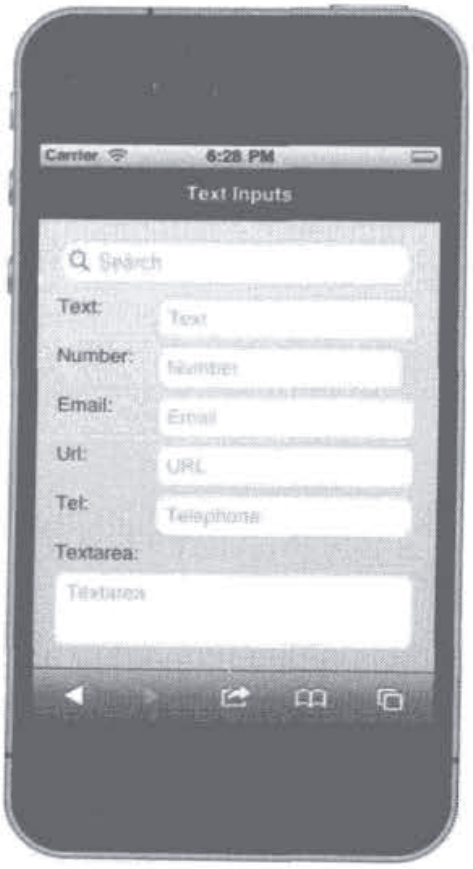


图 4-10 文本输入

从开发人员的角度来看，无需添加任何标记，我们就可以创建 jQuery Mobile 表单和文本输入（见程序清单 4-14）。通过为输入元素添加 data-theme 属性，我们可以为文

[3] 见 http://wave.webaim.org/.

本输入选择一个合适的主题，从而增强表单字段的对比。

程序清单 4-14 文本输入（ch4/text-inputs.html）

```
<input type="text" name="text" value="" id="text" placeholder="Text"/>
<input type="number" name="number" value="" id="number" />
<input type="email" name="email" value="" id="email" data-theme="d" />
<input type="url" name="url" value="" id="url" />
<input type="tel" name="tel" value="" id="tel" />
<input type="search" name="search" value="" id="search" />
<textarea cols="40" rows="8" name="textarea" id="textarea"></textarea>
```

提示：为了以一种可访问的方式来隐藏标签，可以为元素附加 ui-hidden-accessible 样式。例如，在图 4-10 中，我们将该技术应用到搜索字段中。这就可以在保留 508 兼容性的同时，将标签优雅地隐藏起来。

```
<label for="search" class="ui-hidden-accessible">Search</label>
        <input type="search" id="search" placeholder="Search" />
```

在构建表单时，一定要将输入字段与其语义类型关联起来，这很重要。这种关联有两种优势。首先，当输入字段接收到焦点时，它会为用户显示合适的键盘。例如，被指明为 type="number"的字段会自动向用户显示一个数字键盘（见图 4-11）。而使用 type="tel"进行关联的字段，则会显示一个特定的电话号码键盘（见图 4-12）。

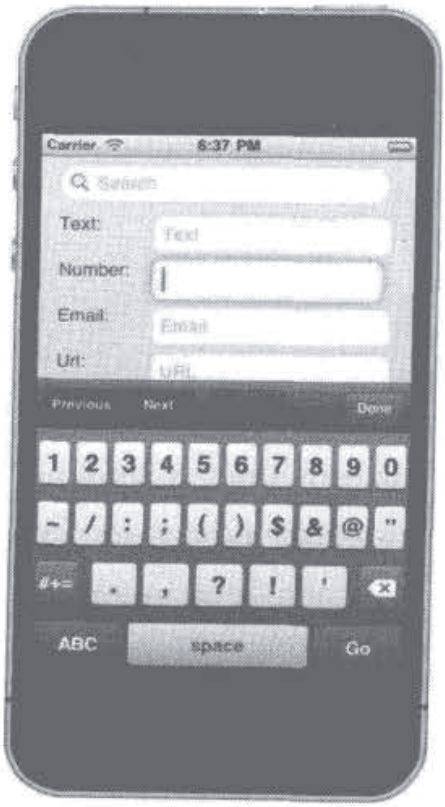


图 4-11 数字键盘

图 4-12 电话号码键盘

此外，该规范允许浏览器针对字段类型应用验证规则。在用户填写表单期间，浏览器能能够自动对每个字段类型进行实时验证。有关移动输入类型和属性的完整列表，请见 Peter-Paul Koch 的“Input tests for mobile”[4]。它列出了所有可用的移动输入类型和属性，以及各种类型的浏览器是否对其提供支持。

所有移动浏览器都能够很好支持的另外一个特性是 placeholder 属性。该属性为文本输入添加了一个提示或标签，而且能够在字段接收到焦点时，自动消失（见程序清单 4-14）。

> **注意：**搜索字段（type="search"）的样式和行为与其他输入类型略微不同。它包含一个左对齐的“搜索”图标，而且它的左右两个圆角呈胶囊形状。当用户输入文本时，则会出现一个右对齐的“删除”图标，用于清除用户的输入。

[4] 见 http://www.quirksmode.org/html5/inputs_mobile.html.

1. 动态文本输入

textinput 插件是一个能够自动增强文本输入和文本区域的微件（widget）。我们可以使用该插件来动态创建、启用和禁用文本输入（见程序清单 4-15）。

程序清单 4-15 textinput 插件示例（ch4/dynmic-text-input.html）

```
// Create text input with markup-driven options
$( '<input type="text" name="text1" value="" data-theme="c" />' )
        .insertAfter( "#firstName" )
        .textinput();

// Create text input with plugin-driven options
$( '<input type="text" name="text2" id="text2" value="" />' )
        .insertAfter( "#text1" )
        .textinput({
                theme: 'c'
        });

// Disable text input
$( "#text1" ).textinput( "disable" );

// Enable text input
$( "#text1" ).textinput( "enable" );
```

2. 文本输入选项

textinput 插件具有如下选项。

initSelector *CSS selector string*

default: "input[type='text'], input[type='search'], :jqmData(type='search'), input[type='number'], :jqmData(type='number'), input[type='password'], input[type='email'], input[type='url'], input[type='tel'], textarea"

initSelector 用来定义用于触发 widget 插件初始化的选择器（元素类型、数据角色[data role]等）。例如，由默认选择器匹配的所有元素可以通过 textinput 插件来增强。要重写该选择器，可以绑定到 mobileinit 事件，并根据需要更新选择器。

```
$( document ).bind( "mobileinit", function(){
        $.mobile.textinput.prototype.options.initSelector = "...";
});
```

theme *string*

default: null。继承自父容器。

为文本元素设置主题调色板配色方案。这是一个取值范围为 a～z 的字母，它映射到你的主题中所包含的调色板。默认情况下，如果没有显式设置主题

则所有的元素都会继承其父容器的同一个调色板颜色。该选项还可以公开作为一个数据属性：data-theme="a"。

```
$( "#text1" ).textinput({ theme: "a" });
```

3. 文本输入方法

textinput 插件具有如下方法。

enable: *enable a disabled textinput or textarea.*

```
$( "textarea" ).textinput( "enable" );
```

disable: *disable a textinput or textarea.*

```
$( "textarea" ).textinput( "disable" );
```

4. 文本输入事件

textinput 插件支持如下事件：

create *triggered when a text input is created*

当创建一个自定义文本输入时，会触发该事件。它并不是用来创建一个自定义输入的。

```
$( '<input type="text" name="text2" id="text2" value="" />' )
        .textinput({
                theme: 'c',
                create: function(event) {
                        console.log( "Creating text input..." );
                }
        })
        .insertAfter( "#text1" );
```

4.2.3 选择菜单

在无需添加额外标记的情况下，jQuery Mobile 框架就能够自动增强所有本地的选择元素（见程序清单 4-16）。

程序清单 4-16 本地的选择菜单（ch4/select-menu-natire.html）

```
<label for="genre">Genre:</label>
<select name="genre" id="genre">
        <option value="action">Action</option>
        <option value="comedy">Comedy</option>
        <option value="drama">Drama</option>
</select>
```

这种转变会使用 jQuery Mobile 风格的按钮来取代原始的选择，而且前者包含一个右对齐的下拉箭头图标。默认情况下，轻敲该选择按钮，会为 OS 启动本地选择选择器（见图 4-13）。作为一种替换方法，我们可以配置 jQuery Mobile，使其显示自定义的选择菜单，下一节会讲到这种方法。

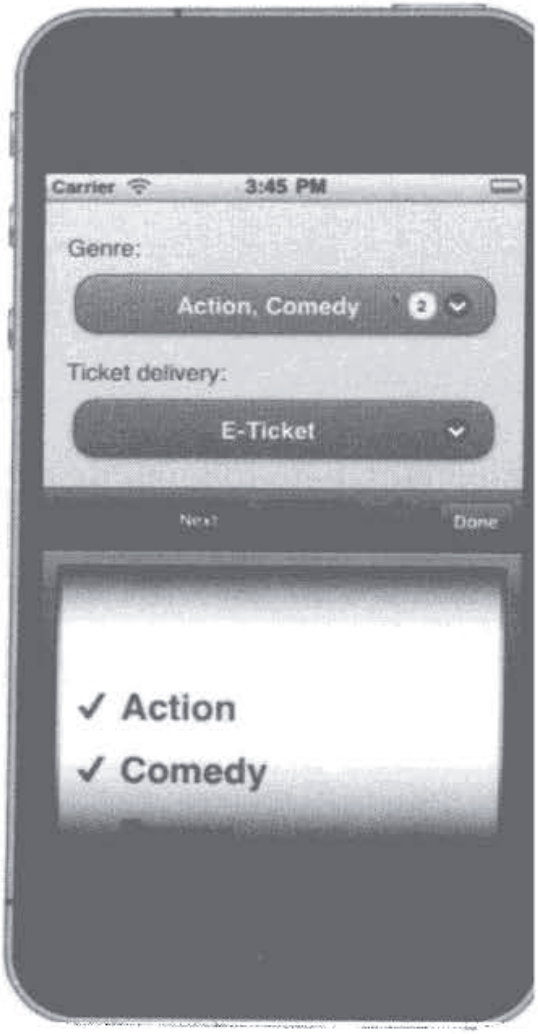


图 4-13 选择菜单

在用户进行选择之后，选择按钮会显示已选定选项的值。如果对按钮来说，文本值太长，则文本将会被截断，并在后面显示一个省略号。此外，在用户选择了多个选项后，多选按钮会对已选中的选项显示计数泡或进行标记（见图 4-13）。这是一个可以用来突出显示已选择选项的数量的视觉效果。

> **警告：** 在使用 multiple="multiple"属性创建选择菜单时，有些移动平台不支持多选特性。在需要使用多选菜单的时候，建议使用自定义菜单。

1. 自定义选择菜单

替代本机呈现选项列表的一个方法是，我们可以使用一个自定义的 HTML/CSS 视图来呈现选择菜单（见图 4-14）。

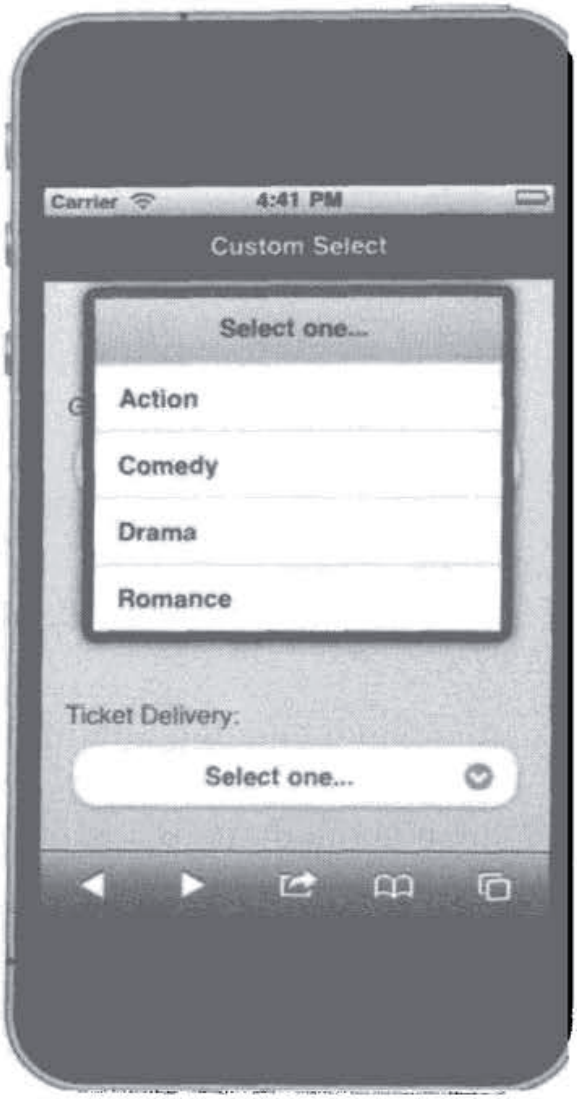


图 4-14 自定义选择菜单

对该视图来说，可以为选择元素添加 data-native-menu="false"属性（见程序清单 4-17）。

程序清单 4-17 自定义选择菜单（ch4/select-menu-custom.html）

```
<label for="genre">Genre:</label>
<select name="genre" id="genre" data-native-menu="false" data-theme="a">
        <option value="">Select one...</option>
        <option value="action">Action</option>
        <option value="comedy">Comedy</option>
        <option value="drama">Drama</option>
</select>
```

以自定义方式呈现选择菜单与本机呈现菜单的优劣对比如下所示。

自定义方式的优势：

- 在所有设备上提供了统一的用户体验。
- 自定义菜单普遍支持多选的选项列表。
- 增加了一种优雅的方式来处理占位符选项。下一节会讲解占位符选项。
- 自定义菜单是可主题化的（见程序清单 4-17）。

自定义方式的劣势：

- 与本机呈现选择菜单相比，性能要差一些。当相比较的菜单中包含许多选项时，这种性能差距会更明显。

> **注意：** 你也可以使用 Mobiscroll[5]插件来自定义选择菜单。在本章末尾会介绍该插件的几个示例，用以演示它在日期选择器和搜索过滤器方面的使用。

2. 占位符选项

对自定义选择菜单来说，占位符是一个独特的特性，它具有如下 3 种好处。

（1）占位符要求用户做出一个选择。默认情况下，如果没有配置占位符，则列表中的第一个选项会被选中。

（2）占位符可以为未选定的选择按钮显示提示文本（见图 4-14）。例如，未选定的 Ticket Delivery 字段将会与占位符文本“Select one...”一起显示。

（3）在显示选项列表时，占位符也可以作为页眉来显示（见图 4-14）。

我们可以用 3 种方式来配置占位符。

（1）为选项添加不带有任何值的文本。

```
<option value="">Select one…</option>
```

（2）在选项包含文本和值的时候，可以为其添加 data-placeholder="true"属性。

```
<option value="null" data-placeholder="true">Select one…</option>
```

[5] 见 http://code.google.com/p/mobiscroll/.

3. 如果需要一个不带有提示文本和页眉的字段，可以使用一个空选项。

```
<option value=""></option>
```

3. 动态选择菜单

selectmenu 插件是一个能自动增强选择菜单的微件。使用该插件，我们能够动态创建、启用、禁用、打开或关闭选择菜单（见程序清单 4-18）。

程序清单 4-18 动态选择菜单（ch4/dynamic-select-menu.html）

```
// Create select menu with markup-driven options
$( '<select name="select1" id="select1" data-theme="e">...</select>' )
        .insertAfter( "#foo" )
        .selectmenu();

// Create select menu with plugin-driven options
$( '<select name="select2" id="select2">...</select>' )
        .insertAfter( "#select1" )
        .selectmenu({
                theme: "e",
                overlayTheme: "c",
                disabled: false,
                nativeMenu: false
        });
```

4. 选择菜单选项

selectmenu 插件具有如下选项。

corners *boolean*

default: true

与其他按钮类型一样，选择菜单按钮在默认情况下也是圆角的。将该选项设置为 false 可以移除圆角。该选项还可以公开作为一个数据属性：data-corners="false"。

```
$( "#select1" ).selectmenu({ corners: false });
```

disabled *boolean*

default: false

禁用该元素。selectmenu 插件也有 enable 和 disable 方法，用来动态启用和禁用控件。

```
$( "#select1" ).selectmenu({ disabled: true });
```

hidePlaceholderMenuItems *boolean*

default: true

默认情况下，当选择菜单打开时，占位符菜单条目是隐藏不见的。为了让

占位符条目是可选择的，将该值设置为 false。

```
$( "#select1" ).selectmenu({ hidePlaceholderMenuItems: false });
```

icon *string*
default: "arrow-d"

设置选择按钮的图标。该选项还可以公开作为一个数据属性：data-icon="plus"。

```
$( "#select1" ).selectmenu({ icon: "plus" });
```

iconpos *string*
default: "right"

设置图标位置。可能的值为"left"、"right"、"none"和"notext"。"notext"值会将选择按钮（select）显示为一个只带有图标的按钮，而且该按钮没有占位符文本。"none"值将会彻底移除图标。该选项还可以公开作为一个数据属性:data-iconpos="none"。

```
$( "#select1" ).selectmenu({ iconpos: "notext" });
```

iconshadow *boolean*
default: true

当该选项的值为 true 时，jQuery Mobile 框架会为图标添加阴影。该选项还可以公开作为一个数据属性：data-iconshadow="false"。

```
$( "#select1" ).selectmenu({ iconshadow: false });
```

initSelector *CSS selector string*
default: "select:not(:jqmData(role='slider'))"

initSelector 定义用来触发 widget 插件自动初始化的选择器（元素类型、数据角色[data role]等）。例如，由默认选择器匹配的所有元素都会被 selectmenu 插件增强。为了重写该选择器，可以绑定到 mobileinit 事件，并根据情况更新选择器。

```
$( document ).bind( "mobileinit", function(){
        $.mobile.selectmenu.prototype.options.initSelector = "..";
});
```

inline *boolean*
default: false

如果该选项设置为 true，则会让选择按钮以内嵌（inline）按钮的形式显示。默认情况下，选择按钮会占据其容器的整个宽度。与之相对比的是，内嵌按钮只占据其占位符文本的宽度。该选项还可以公开作为一个数据属性：data-inline="true"。

```
$( "#select1" ).selectmenu({ inline: true });
```

nativeMenu *boolean*
default: true

默认情况下，选择按钮会为 OS 启动本地的选择选择器（select picker）。要以自定义的 HTML/CSS 视图来呈现选择菜单，需要将该值设置为 false。该选项还可以公开作为一个数据属性：data-native="false"。

```
$( "#select1" ).selectmenu({ nativeMenu: false });
```

shadow *boolean*
default: true

默认情况下，选择按钮会应用阴影。将该选项设置为 false 则会移除阴影。该选项还可以公开作为一个数据属性：data-shadow="false"。

```
$( "#select1" ).selectmenu({ shadow: false });
```

theme *string*
default: null. Inherited from parent.

为元素设置主题调色板配色方案。这是一个取值范围为 a～z 的字母，它映射到你的主题中所包含的调色板。默认情况下，元素会继承其父容器的同一个调色板颜色。该选项还可以公开作为一个数据属性：data-theme="a"。

```
$( "#select1" ).selectmenu({ theme: "a" });
```

5. 选择菜单方法

selectmenu 插件具有如下方法。

enable：启用一个被禁用的选择按钮。

```
$( "#select1" ).selectmenu( "enable" );
```

disable：禁用一个选择按钮。

```
$( "#select1" ).selectmenu( "disable" );
```

open：打开一个关闭的选择按钮。该方法只能用于自定义选择。

```
$("#select1").selectmenu("open");
```

close：关闭一个打开的选择按钮。该方法只能用于自定义选择。

```
$("#select1").selectmenu("close");
```

refresh：更新自定义的选择菜单。

该方法会更新自定义的选择菜单，以反映本地的选择元素的值。例如.

如果本地选择的 selectedIndex 被更新，我们可以调用“refresh”方法来重新构建自定义选择。如果传递了一个 true 参数，可以强制进行更新并重新构建自定义选择。

```
// Select the third menu option and refresh the menu
        var myselect = $( "#select1" );
        myselect[0].selectedIndex = 2;
        myselect.selectmenu( "refresh" );

        // refresh and rebuild the list
        myselect.selectmenu( "refresh", true );
```

6. 选择菜单事件

selectmenu 插件支持如下事件：

create：在创建一个选择菜单时触发该事件。

该事件在创建一个选择菜单时触发。它并不是用来创建一个自定义元素。

```
$( '<select name="select2" id="select2">...</select>' )
        .insertAfter( "#select1" )
        .selectmenu({
                create: function(event) {
                        console.log( "Creating select menu..." );
                }
        });
```

4.2.4 单选按钮

单选按钮只允许用户选择一个条目（见图 4-15）。例如，在从多个应用程序设置选项中选择一个设置时，通常会使用单选按钮来实现，原因是单选按钮比较简单而且易于使用。用户可以通过轻敲单选按钮来完成他们的选择，jQuery Mobile 会自动更新底层的表单控件。

在程序清单 4-19 中，我们添加了 3 个额外的属性，以帮助设计和放置单选按钮。第一个属性 data-role="controlgroup"对按钮进行编组，而且编组后的按钮是圆角的。第二个属性 data-type="horizontal"重写按钮默认的垂直定位，以水平方式显示按钮。最后一个属性用来对按钮进行主题化。默认情况下，单选按钮会继承其父控件的主题。但是，如果想为单选按钮应用其他主题，这需要为相应单选按钮的标签添加 data-theme 属性。

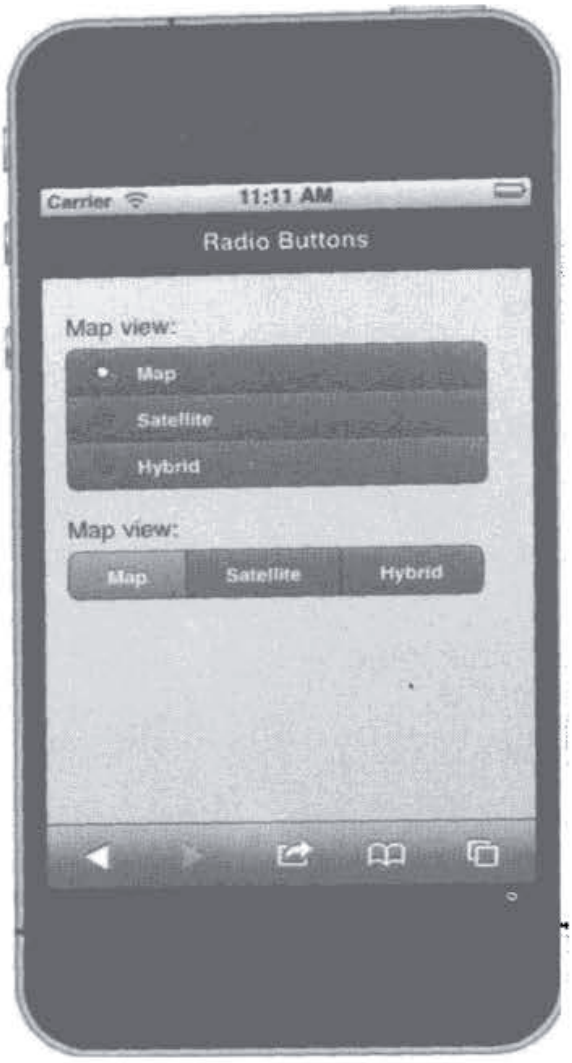


图 4-15 单选按钮

程序清单 4-19 水平放置的单选按钮（ch4/radio-buttons.html）

```
<fieldset data-role="controlgroup" data-type="horizontal">
    <legend>Map view:</legend>
    <input type="radio" name="map" id="map1" value="Map" />
    <label for="map1" data-theme="b">Map</label>

    <input type="radio" name="map" id="map2" value="Satellite" />
    <label for="map2" data-theme="b">Satellite</label>

    <input type="radio" name="map" id="map3" value="Hybrid"  />
    <label for="map3" data-theme="b">Hybrid</label>
</fieldset>
```

警告：如果水平放置的单选按钮的容器无法在一行内显示所有的单选按钮，则按钮会发生重叠现象。为了避免重叠，可以减小按钮的字体大小:ui-controlgroup-horizontal.ui- radio label{ font-size:13px !important;}

1. 动态单选按钮

checkboxradio 插件是一个可重用的微件，能够自动增强单选按钮和复选框。通过该插件，我们可以动态创建、启用、禁用和刷新单选按钮（见程序清单 4-20）。

程序清单 4-20 动态单选按钮（ch4/dynamic-radio-buttons.html）

```
// Create radio buttons with markup-driven options
$( '<fieldset data-role="controlgroup">
        <legend>Map view:</legend>
        <input type="radio" name="map" id="map1" value="Map" />
        <label for="map1" data-theme="c">Map</label>...</fieldset>')
        .insertAfter( "#radio1" );
$.mobile.pageContainer.trigger( "create" );

// Create radio buttons with plugin-driven options
$( '<fieldset data-role="controlgroup">
        <legend>Map view:</legend>
        <input type="radio" name="map" id="map1" value="Map" />
        <label for="map1">Map</label>...</fieldset>' )
        .insertAfter( "#radio1" );
$( "#map1" ).checkboxradio({ theme: "e" });
$( "#map2" ).checkboxradio({ theme: "e" });
$.mobile.pageContainer.trigger( "create" );
```

2. 复选框和单选按钮事件

checkboxradio 插件具有如下选项。

initSelector *CSS selector string*
default: "input[type='checkbox'],input[type='radio']"

initSelector 用于定义用来触发 widget 插件自动初始化的选择器（元素类型、数据角色[data role]等）。例如，由默认选择器匹配的所有元素都会被 checkboxradio 插件增强。要重写该选择器，可以绑定到 mobileinit 事件，然后根据情况更新选择器。

```
$( document ).bind( "mobileinit", function(){
    $.mobile.checkboxradio.prototype.options.initSelector = "...";
});
```

theme *string*
default: null. Inherited from parent.

为复选框或单选按钮设置主题调色板配色方案。这是一个取值范围为 a～z 的字母，它映射到你的主题中所包含的调色板。默认情况下，它会继承其父容器的同一个调色板颜色。该选项还可以公开作为一个数据属性：data-theme="a"。

```
$( "#element1" ).checkboxradio({ theme: "a" });
```

3. 复选框和单选按钮的方法

checkboxradio 插件具有如下方法。

enable: 启用一个被禁用的复选框或单选按钮。

```
$("#element1" ).checkboxradio( "enable" );
```

disable: 禁用一个复选框或单选按钮。

```
$( "#element1" ).checkboxradio( "disable" );
```

refresh:更新自定义的复选框或单选按钮。

该方法用来更新自定义的复选框或单选按钮，以反映本地元素的值。例如，我们可以动态选中一个单选按钮，然后调用“refresh”方法来重建增强的控件。

```
// Dynamically set a checkbox or radio element and refresh it.
$( "#elem1" ).attr( "checked", true ).checkboxradio( "refresh" );
```

4. 复选框和单选按钮的事件

checkboxradio 插件支持如下事件。

create：在创建一个复选框或单选按钮时触发该事件。

在创建一个复选框或单选按钮时触发该事件。它不是用来创建一个自定义元素。

```
$( '#element1' )
        .checkboxradio({
                theme: "e",
                create: function(event) {
                        console.log( "Creating new element..." );
                }
});
```

4.2.5 复选框

复选框是一个常见的表单控件，允许用户从一系列选择中选择多个值（见图 4-16）。用户可以轻敲复选框按钮完成自己的选择，jQuery Mobile 会自动更新底层的表单控件。

用于设计和定位复选框的标记与之前用于单选按钮的标记相同（见程序清单 4-21）。我们仍然添加了三个额外的属性，以帮助设计和放置复选框。第一个属性 data-role=“controlgroup”将复选框元素进行编组，而且编组后的复选框是圆角的。第二

个属性 data-type="horizontal"重写按钮默认的垂直定位，以水平方式显示按钮。最后一个属性用来对按钮进行主题化。默认情况下，复选框会继承其父控件的主题。但是，如果想为复选框应用其他主题，需要为相应复选框的标签添加 data-theme 属性。

图 4-16 复选框

程序清单 4-21 水平放置的复选框（ch4/checkboxes.html）

```
<fieldset data-role="controlgroup" data-type="horizontal">
        <legend>Genre:</legend>
        <input type="checkbox" name="genre" id="c1" />
       <label for="c1" data-theme="c">Action</label>

       <input type="checkbox" name="genre" id="c2" />
       <label for="c2" data-theme="c">Comedy</label>

       <input type="checkbox" name="genre" id="c3" />
        <label for="c3" data-theme="c">Drama</label>
</fieldset>
```

警告：如果水平放置的复选框的容器无法在一行内显示所有的复选框，则复选框会发生重叠现象。为了避免重叠，可以减小复选框的字体大小:ui-controlgroup-horizontal.ui-checkbox label{ font-size:11px !important;}

动态复选框

checkboxradio 插件是一个能够自动增强复选框和单选按钮的微件。通过该插件，我们可以动态创建、启用、禁用和刷新复选框（见程序清单 4-22）。checkboxradio 插件的选项、方法、事件等详细内容请见“动态单选按钮”一节。jQuery Mobile 中，相同的 API 也可以多次使用于单选按钮和复选框。

程序清单 4-22 动态复选框（ch4/dynamic-checkboxes.html）

```
// Create checkboxes with markup-driven options
$( '<fieldset data-role="controlgroup">
        <legend>Genre:</legend>
        <input type="checkbox" name="genre" id="c1" />
        <label for="c1" data-theme="c">Action</label>...</fieldset>' )
        .insertAfter( "#element1" );
$.mobile.pageContainer.trigger( "create" );

// Create checkboxes with plugin-driven options
$( '<fieldset data-role="controlgroup">
        <legend>Genre:</legend>
        <input type="checkbox" name="genre" id="c3" />
        <label for="c3">Action</label>...</fieldset>' )
        .insertAfter( "#create-cb2" );
$( '#c3' ).checkboxradio({ theme: "e" });
$( '#c4' ).checkboxradio({ theme: "e" });
$.mobile.pageContainer.trigger( "create" );
```

4.2.6 滑动条

滑动条是一个常见的表单控件，允许用户在最小范围和最大范围之间选择一个值（见图 4-17）。

例如，在我们的例子中，我们可以使用滑动条在最小和最大设置之间调整音量或屏幕亮度。我们可以调整滑动条的最小和最大边界，也可以设置滑动条的默认值。用户可以通过滑动控件的方式，或者是在滑动条相应的文本字段中输入一个值的方式，来调整滑动条。如程序清单 4-23 所示，对 jQuery Mobile 来说，没有必要添加任何标记就可以增强滑动条。带有 type="range"的任何输入元素都会被自动优化。

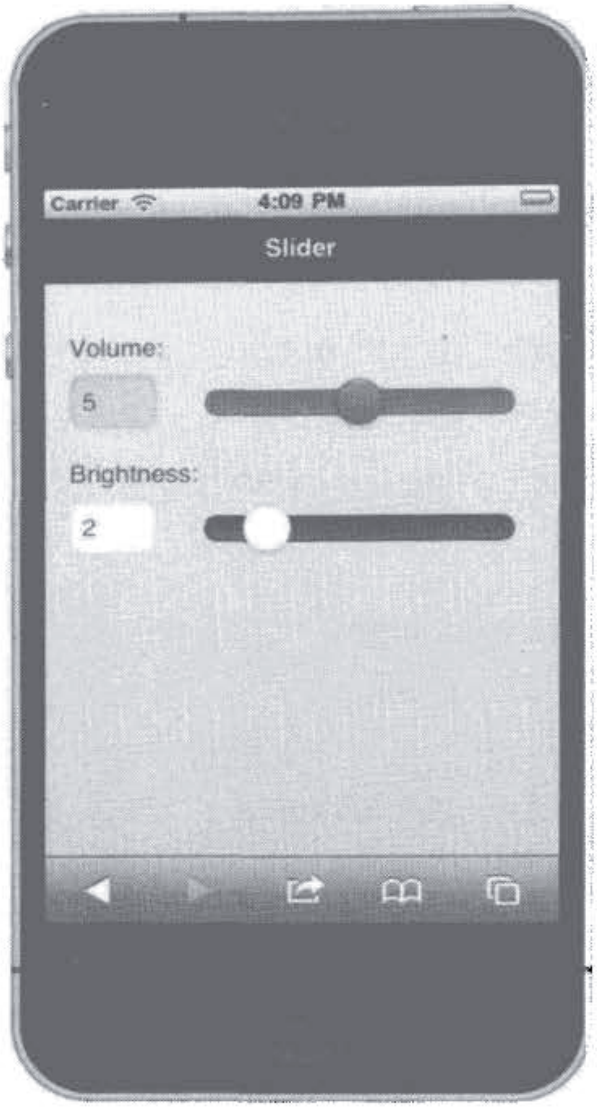


图 4-17 滑动条

程序清单 4-23 滑动条（ch4/slider.html）

```
<label for="volume">Volume:</label>
<input type="range" name="volume" id="volume" value="5" min="0" max="9"/>
```

滑动条包含两个可主题化的组件。其中一个是名为滑动条的前景组件，另外一个是名为轨道的背景组件。这两个组件可以分别进行主题化。为了对滑动条进行主题化，需要为 input 元素添加 data-theme="a"属性。此外，要对轨道进行主题化，需要为 input 元素添加 data-track-theme="a"属性。

```
<input type="range" name="brightness" id="brightness" min="0" max="10" data-theme="b"
data-track-theme="a" />
```

1. 动态滑动条

slider 插件是一个多用途的微件，能够自动增强滑动条和开关控件。通过该插件，

我们可以动态创建、启用、禁用和开/关开关控件（见程序清单 4-24）。

程序清单 4-24 动态滑动条（ch4/dynamic-slider.html）

```
// Create slider with markup-driven options
$( '<label for="s1">Brightness:</label>
 <input type="range" name="s1" id="s1" min="0" max="9" data-theme="d"/>')
        .insertAfter( "#element1" );
$( "#s1" ).slider();

// Create slider with plugin-driven options
$( '<label for="s1">Brightness:</label>
    <input type="range" name="s1" id="s1" min="0" max="10" />' )
        .insertAfter( "#create-s2" );
$( "#s1" ).slider({
        theme: "d",
        trackTheme: "a",
        disabled: false
});
```

2. 滑动条选项

slider 插件具有如下选项。

disabled *boolean*

default: false

禁用控件。slider 插件也有 enable 和 disable 方法，用来动态启用或禁用控件。

```
$( "#element1" ).slider({ disabled: true });
```

initSelector *CSS selector string*

default: "input[type='range'], :jqmData(type='range'), :jqmData(role='slider')"

initSelector 用来定义用于触发 widget 插件自动初始化的选择器（元素类型、数据角色[data role]等）。例如，由默认选择器匹配的所有元素都可以由 slider 插件来增强。要重写该选择器，可以绑定到 mobileinit 事件，并根据情况升级选择器。

```
$( document ).bind( "mobileinit", function(){
        $.mobile.slider.prototype.options.initSelector = "...";
});
```

theme *string*

default : null。继承自父容器。

为滑动条设置主题调色板配色方案。这是一个取值范围为 a～z 的字母，它映射到你的主题中所包含的调色板。默认情况下，滑动条会继承其父容器的

同一个调色板颜色。该选项还可以公开作为一个数据属性：data-theme="a"。

```
$( "#element1" ).slider({ theme: "a" });
```

trackTheme string

default：null。继承自父容器。

为滑动条的轨道设置主题调色板配色方案。这是一个取值范围为a～z的字母，它映射到你的主题中所包含的调色板。默认情况下，轨道会继承其父容器的同一个调色板颜色。该选项还可以公开作为一个数据属性：data-track-theme="a"。

```
$( "#element1" ).slider({ trackTheme: "a" });
```

3. 滑动条方法

slider插件具有如下方法。

enable：启用一个被禁用的滑动条或开关控件。

```
$("#element1" ).slider( "enable" );
```

disable：禁用一个滑动条或开关控件。

```
$( "#element1" ).slider( "disable" );
```

refresh：更新一个自定义的滑动条或开关控件。

该方法会更新自定义滑动条或开关，以反映本地元素的值。例如，我们能够动态更新开关或滑动条，并调用"refresh"方法来重建控件。

```
// Set the switch to "on" and refresh it
var switch = $( "#switch1" );
switch[0].selectedIndex = 1;
switch.slider( "refresh" );

// Maximize the slider's volume control and refresh it
$( "#volume" ).val( 10 ).slider( "refresh" );
```

4. 滑动条事件

slider创建支持如下事件。

create：在创建一个滑动条或开关控件时被触发。

该事件在在创建一个滑动条或开关控件时被触发。它不是用来创建自定义

元素的。

```
$( '<select name="switch2" id="switch2">...</select>' )
        .insertAfter( "#element1" )
        .slider({
                create: function(event) {
                        console.log( "Creating new element..." );
                });
```

4.2.7 开关控件

开关控件（见图 4-18）通常用来管理布尔值的 on/off 标记。

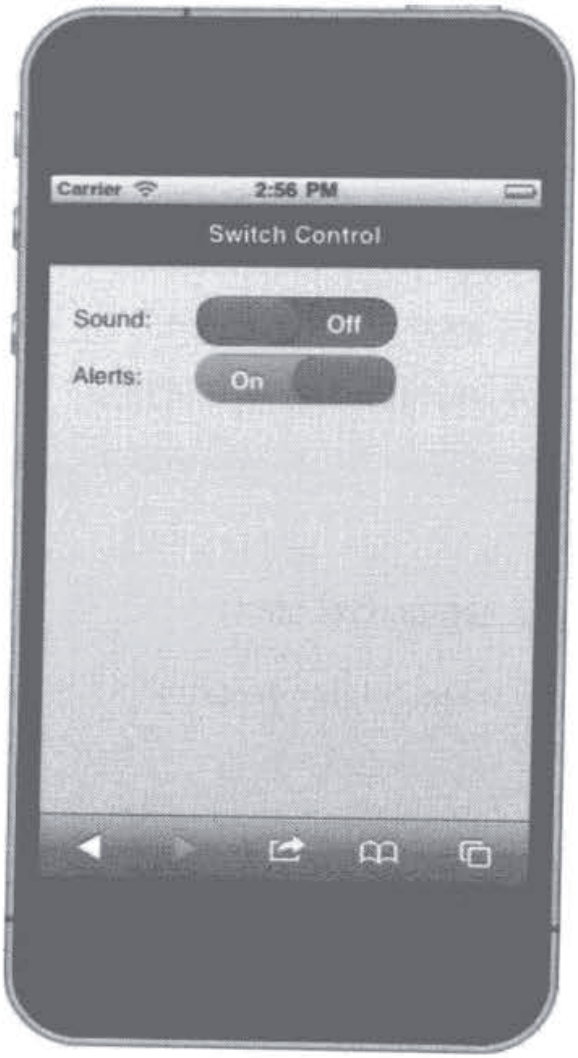


图 4-18 开关控件

例如，由于开关控件具备简单和易于使用的特点，用户在操作应用程序的设置时，会优先选用开关控件。要切换开关，用户可以轻敲该控件，也可以滑动开关控件。要创建一个开关控件，添加一个选择元素，而且该选择元素带有 data-role="slider"属性

然后再添加两个选项，用来管理 on/off 状态（见程序清单 4-25）。

程序清单 4-25 开关控件（ch4/switch-control.html）

```
<label for="alerts">Alerts:</label>
<select name="slider" id="alerts" data-role="slider">
        <option value="off">Off</option>
        <option value="on">On</option>
</select>
```

开关控件也包含两个可主题化的组件。其中一个是名为滑动条的前景组件，另外一个是名为轨道的背景组件。这两个组件可以分别进行主题化。为了对滑动条进行主题化，需要为 select 元素添加 data-theme="a"属性。此外，要对轨道进行主题化，需要为 select 元素添加 data-track-theme="a"属性。

```
<select name="slider" data-theme="b" data-track-theme="c" data-role="slider">
        <option value="off">Off</option>
        <option value="on">On</option>
</select>
```

动态开关控件

前面讲到，slider 插件是一个能够自动增强开关控件的微件。通过该控件，我们可以动态创建、启用、禁用和开/关开关控件（见程序清单 4-26）。slider 插件的选项、方法、事件等详细内容请见“动态滑动条”一节。jQuery Mobile 中，相同的 API 也可以多次使用于滑动条和开关控件。

程序清单 4-26 动态开关控件（ch4/dynamic-swnitch-control.html）

```
// Create switch with markup-driven options
$( '<select name="switch1" data-role="slider" data-theme="c"></select>' )
        .insertAfter( "#foo" )
        .slider();

// Create switch with plugin-driven options
$( '<select name="switch2" id="switch2">...</select>' )
        .insertAfter( "#switch1" )
        .slider({
                theme: "b",
                trackTheme: "c",
                disabled: false
        });
```

4.2.8 本地表单元素

jQuery Mobile 能够自动增强页面内的所有表单元素。然而，如果你想回退到（fall back

to）本地控件（见图 4-18），则可以在全局或者字段级别上进行配置（见程序清单 4-27）。

图 4-19 本地表单元素

程序清单 4-27 本地表单元素（ch4/native.html）

```
// Selectively choose which elements are native with data-role="none"
<label for="name">Text Input</label>
<input type="text" name="name" id="name" value="" data-role="none" />

<label for="slider2">Switch:</label>
<select name="slider2" id="slider2" data-role="none">
        <option value="off">Off</option>
        <option value="on">On</option>
</select>

// Globally configure native elements by selector
$(document).bind('mobileinit',function(){
        $.mobile.page.prototype.options.keepNative = "input, select";
});
```

为了分别设置表单字段，以显示其本地控件，可以为其元素添加 data-role="none" 属性。另外一种方法是，在 mobileinit 事件初始化时，通过设置 keepNative 选择器 [1]

全局方式配置应该以本地方式呈现的表单元素。例如，在程序清单 4-27 中，我们对选择器进行了配置，使其能够在本地外观中显示所有的 input 和 select 元素。第 8 章会详细讲解如何配置 jQuery Mobile。

HTML5 提供了多个新的输入类型，以帮助收集数据和时间输入。现在我们有 time、date、month、week、datetime 和 datetime-local 输入类型（见程序清单 4-28）。

程序清单 4-28 HTML5 的时间、日期类型（ch4/dates.html）

```
<input type="time" name="time" />
<input type="datetime-local" name="dtl" />
<input type="date" name="date" />
<input type="month" name="month" />
<input type="week" name="week" />
<input type="datetime" name="dt" />
```

是否支持这些新的 HTML5 输入类型，则取决于用户所使用的浏览器（见 http://www.quirksmode.org/html5/inputs.html）。支持这些特性的较新的浏览器能够显示有用的日期选择器（见图 4-20），而不支持这些特性的浏览器则会回退到文本输入。

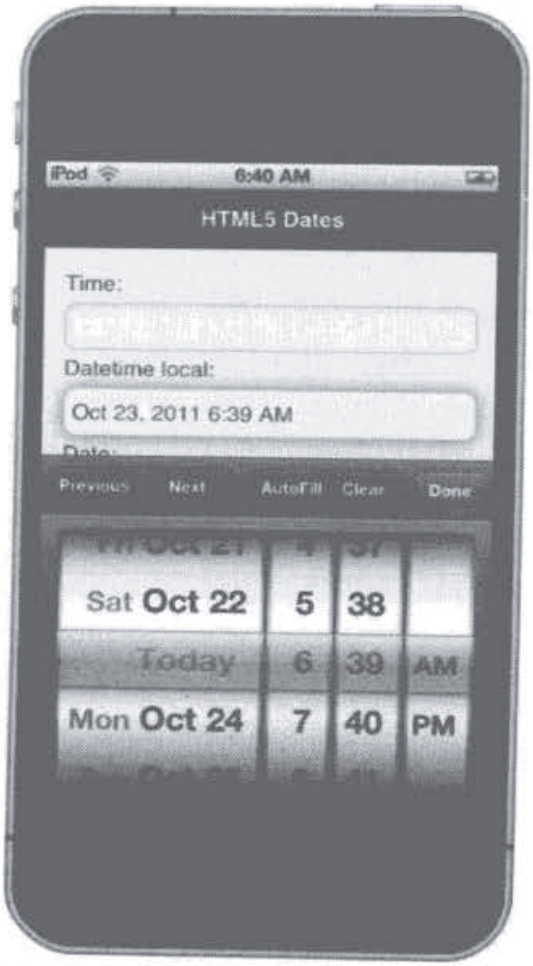


图 4-20 HTML5 的日期

4.2.9 Mobiscroll 日期选择器

Mobiscroll[6]是一个优化的日期选择器，用于触摸屏设备。Mobiscroll API 是可配置的[7]，可以允许显示多个日期和时间的组合（见图 4-21）。此外，Mobiscroll 是可主题化的，也可以进行自定义，以显示任何需要的数据（见图 4-22）。

图 4-21 Mobiscroll 日期选择器

6 见 http://code.google.com/p/mobiscroll/.
7 见 http://code.google.com/p/mobiscroll/wiki/Documentation.

图 4-22 带有自定义列表的 Mobiscroll

例如，我们可以更新 Mobiscroll 的选项，以创建一个自定义的电影搜索（见程序清单 4-29）。Mobiscrol 插件也是一个非常灵活的控件，可以用于许多不同的使用案例。

程序清单 4-29 Mobiscroll（ch4/mobiscroll.html）

```
// Import the Mobiscroll resources
<script type="text/javascript" src="jquery.scroller-1.0.2.js"></script>
<link type="text/css" rel="stylesheet" href="jquery.scroller-1.0.2.css"/>

// Display the default date picker (see Figure 4-21).
$( "#element1" ).scroller();

// Display a custom filter for a movie search (see Figure 4-22).
$( "#element2" ).scroller({
        setText: 'Search',
        theme: 'sense-ui',
        wheels: [{
```

```
        wheels: [{
                'Rating': { '5-star': '*****', '4-star': '****' ... },
                'Genre': { 'action': 'Action', 'comedy': 'Comedy', ...},
                'Screen': { '3d': '3D', 'imax': 'IMAX', 'wide': 'Wide' }
        }]
});
```

4.3 总结

在本章，我们讲解了每一个标准的 HTML 表单组件，以及 jQuery Mobile 如何自动增强每一个组件，以在所有设备上提供统一的用户界面。

在讲解每一个组件时，我们也讨论了其用途，并了解了每一种组件的常见使用案例。我们还讲解了对每个表单元素而言是唯一的 jQuery Mobile 数据属性，并查看了如何通过修改这些属性以配置和设计表单的代码示例。此外，我们还讲解了与每个表单组件相关的插件，以及当用户需要一个更为动态的体验时，如何使用插件 API 来动态创建、增强和更新我们自己的组件。

最后，我们探究了 Mobiscroll 插件的特性，以及它如何为日期选择器、搜索过滤器和自定义列表提供了简洁而且灵活的界面。

在第 5 章，我们的关注重点将从收集用户信息转移到呈现用户信息。尤其是，我们会讲解用来设计和配置信息列表的多种方式。

第5章

列表视图

列表是一种广受欢迎的用户界面组件，原因是通过它们用户能够简单而且有效地进行浏览体验。列表也是一种能够以多种方式进行设计的灵活组件，能够很好地适应不同的屏幕尺寸。我们无论是浏览邮件、通讯录、音乐还是查看设置，这些应用程序都以一种略微不同的样式风格来显示一系列信息。无论是只包含文本的基本列表，还是带有图形和详细元数据的复杂列表，都必须足够灵活，以支持多种配置。幸运的是，jQuery Mobile 支持所有这些列表的配置甚至更多。在本章，我们将会讲解在 jQuery Mobile 内设计（style）和配置列表的细节，以及如何为列表添加搜索过滤器。最后，我们还会讲解列表视图插件 API，并会看到几个动态创建和更新列表的示例。

5.1 列表基础知识

当为列表元素添加了 data-role="list" 属性之后，jQuery Mobile 能够将任何本地 HTML 列表（<ul>或<ol>）自动增强为一个优化的移动视图。在默认情况下，增强后的列表在显示时，会占据整个屏幕，如果列表条目包含链接，则会以易于触摸的按钮方式来显示，而且会带有一个右对齐的箭头图标（见图 5-1，产生该效果的代码见程序清单 5-1）。默认情况下，列表会使用调色板颜色“c”（灰色）来样式化。要应用其他主题，则需要为列表元素或列表条目（<li>）添加 data-theme 属性。

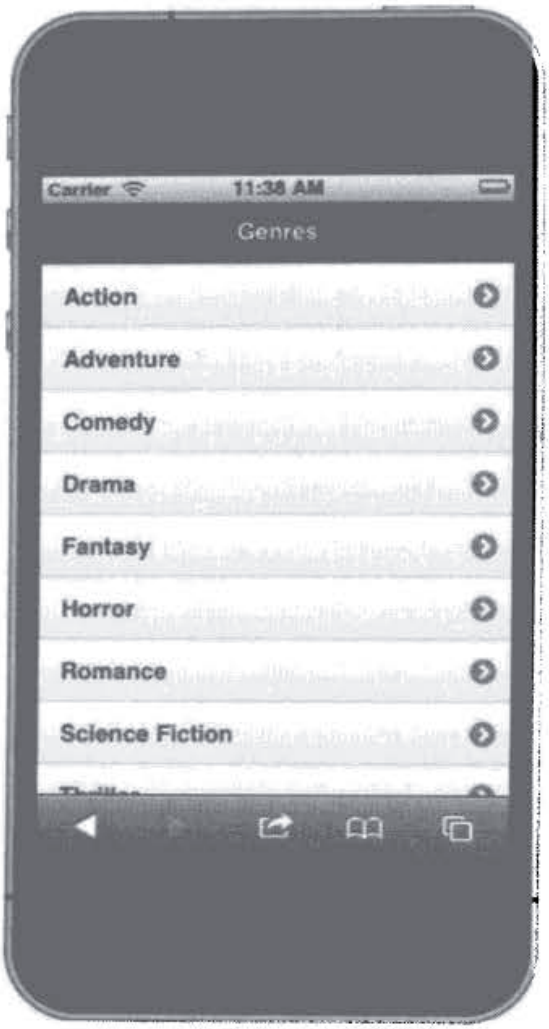


图 5-1 基本列表

程序清单 5-1 基本列表（ch5/list-basic.html）

```
<ul data-role="listview" data-theme="c">
        <li><a href="#">Action</a></li>
        <li><a href="#">Adventure</a></li>
        <li><a href="#">Comedy</a></li>
</ul>
```

5.2 内置列表

内置列表（inset list）在显示时，不会占据整个屏幕。相反，它会自动存在于带有圆角的区域块内部，而且具有额外空间的边距设置。要创建一个内置列表，需要为列表元素添加 data-inset=“true“属性（见图 5-2 和图 5-3，相关代码见程序清单 5-2）。

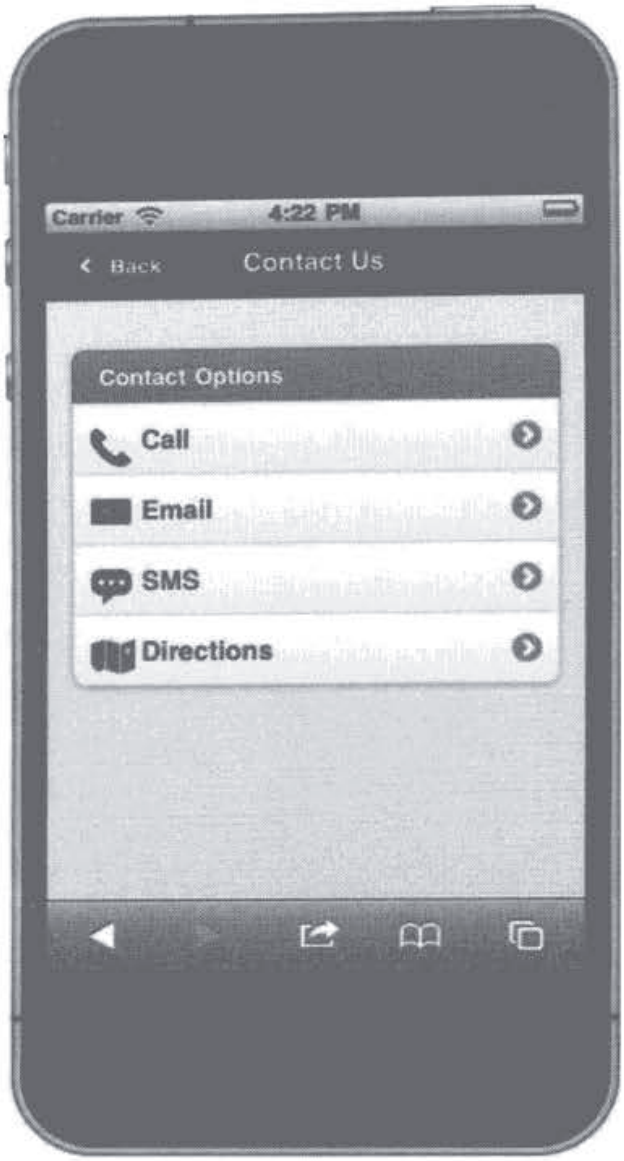


图 5-2 内置列表（iOS）

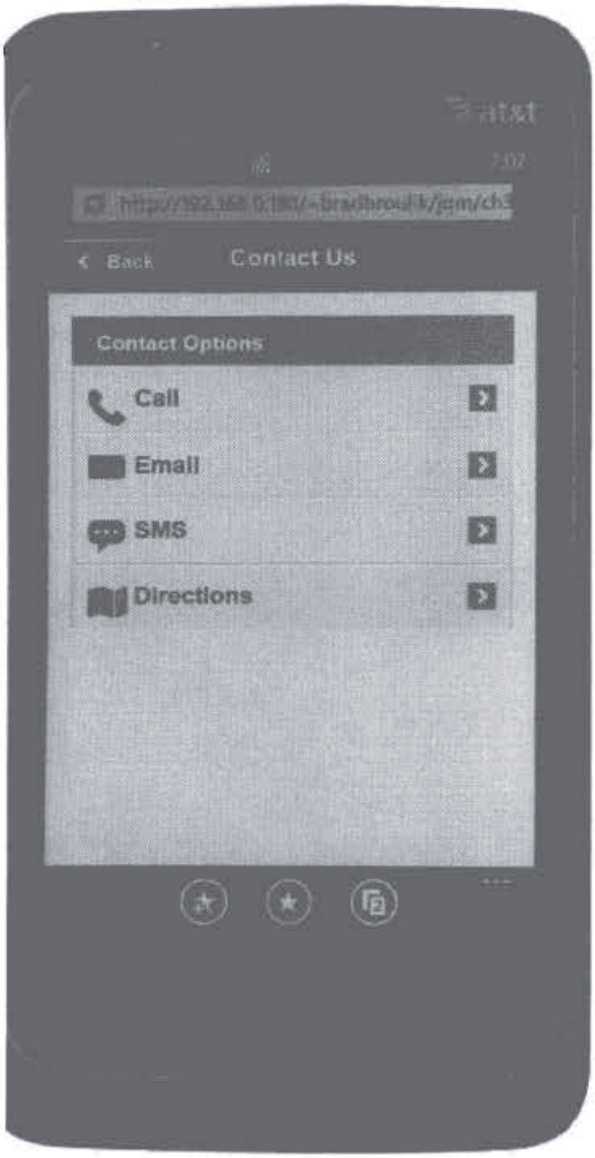


图 5-3 内置列表（Windows Phone 7）

程序清单 5-2 内置列表（ch5/list-inset.html）

```
<ul data-role="listview" data-inset="true">
    <li data-role="list-divider">Contact Options</li>
    <li><a href="#"><img src="phone.png" class="ui-li-icon">Call</a></li>
    <li>...</li>
</ul>
```

5.3 列表分割线

列表分割线（list divider）可以用作一组列表条目的页眉。例如，如果我们的应用程序有一个日历列表，我们可以选择按照日期对日历事件进行分组（见图 5-4）。列表分割线也可以用作内置列表的页眉。在上一个例子中，我们使用列表分割线设置了内

置列表的页眉（见图 5-2 和程序清单 5-2）。

为了创建列表分割线，需要为任何列表条目添加 data-role="list-divider"属性。列表分割线的默认文本在显示时是左对齐的。

> **提示：** 在图 5-4 中，列表条目同时包含左对齐和右对齐的文本。要让文本以右对齐方式放置，需要使用一个包含 ui-li-aside 类的元素对其进行包装（见程序清单 5-3）。

默认情况下，列表分割线使用调色板颜色“b”（浅蓝色）进行样式化。要应用其他主题，则需要为列表元素添加 data-divider-theme="a"属性。

图 5-4 列表分割线

程序清单 5-3 列表分割线（ch5/list-dividers.html）

```
<ul data-role="listview">
  <li data-role="list-divider" data-divider-theme="a">
     Mon <p class="ui-li-aside">Feb 6 2012</p>
  </li>
  <li>
     <a href="#">6 PM <span class="ui-li-aside">Birthday Party</span></a>
  </li>
</ul>
```

5.4 带有缩略图和图标的列表

通过将一个图像作为列表条目的第一个子元素添加到列表条目中，可以在屏幕左方为列表条目添加缩略图（见图 5-5，相关代码见程序清单 5-4）。jQuery Mobile 框架会将图像缩放为 80 像素的正方形。

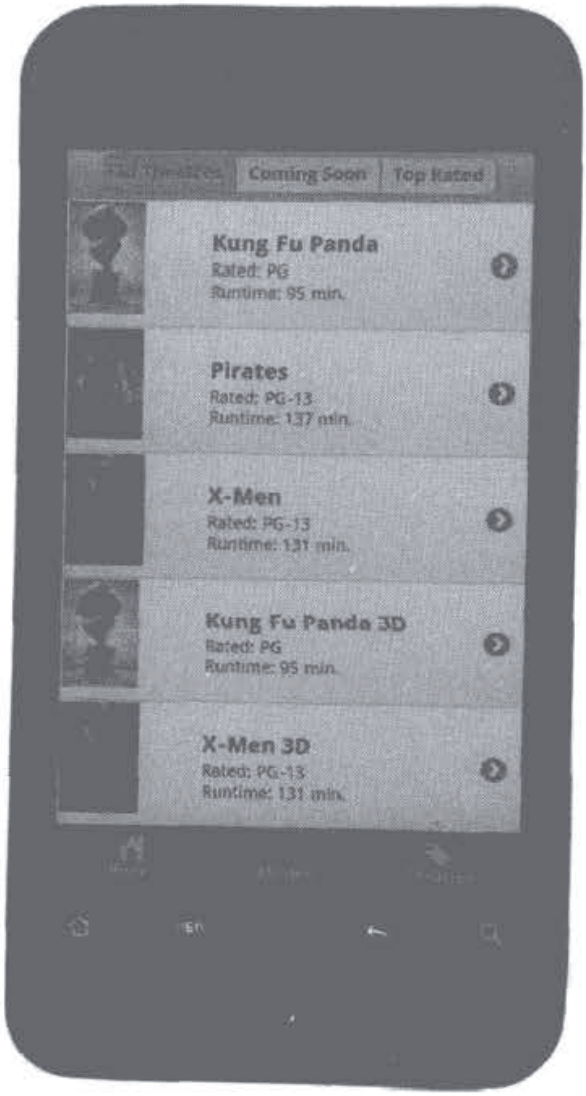


图 5-5 带有缩略图的列表

程序清单 5-4 带有缩略图的列表（ch5/list-thumbnails.html）

```
<ul data-role="listview">
    <li>
        <a href="movies/kung-fu-panda.html">
            <img src="images/kungfupanda2.jpg" />
            <h3>Kung Fu Panda</h3>
            <p>Rated: PG</p>
            <p>Runtime: 95 min.</p>
        </a>
    </li>
    ...
</ul>
```

我们也可以使用更小的图标来取代缩略图。要在列表条目中使用 16×16 像素的图标，需要为图像元素添加 ui-li-icon 类（见图 5-6，相关代码见程序清单 5-5）。

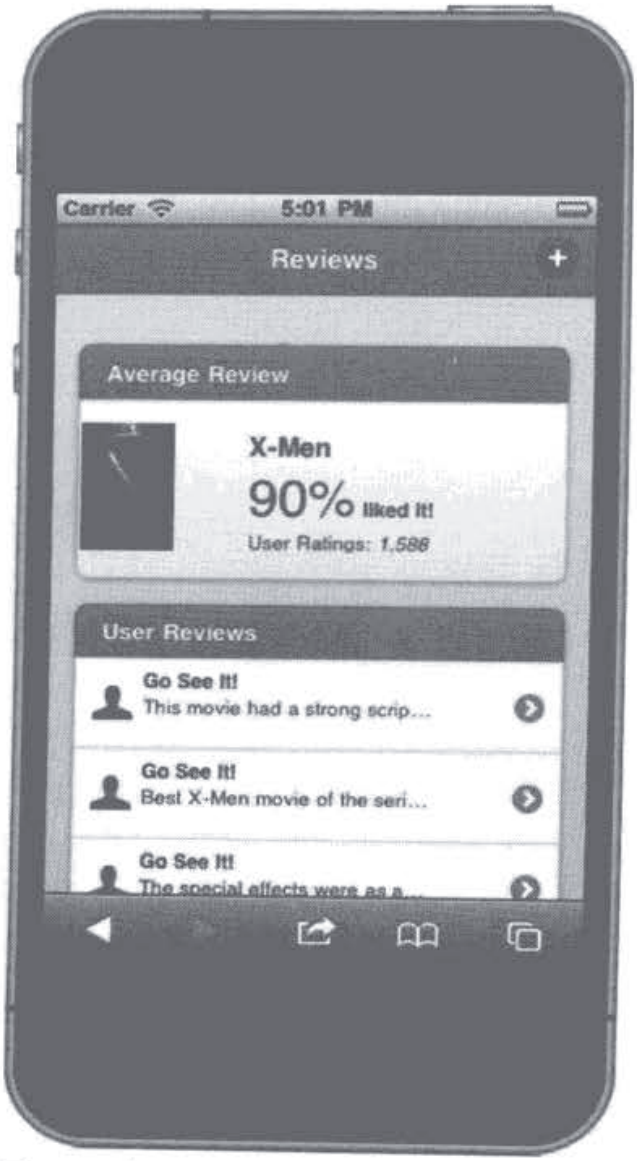


图 5-6 带有图标的列表

程序清单 5-5 带有图标的列表（ch5/list-icons.html）

```
<ul data-role="listview" data-inset="true" data-theme="d">
        <li data-role="list-divider">User Reviews</li>
        <li>
            <a href="reviews/xmen/404.html">
                <img src="images/111-user.png" class="ui-li-icon">
                <p><strong>Go See It!</strong></p>
                <p>This movie had a strong script and ... </p>
            </a>
        </li>
        ...
</ul>
```

5.5 拆分按钮列表

在某些情况下，我们需要让每个列表条目支持多个动作，对此，可以创建具有主（primary）按钮和附属（secondary）按钮的拆分按钮列表。例如，我们可以修改最初的电影列表示例，使其支持多个动作。我们的主按钮会继续显示电影详情，而新的附属按钮可以用来购买电影票（见图 5-7）。

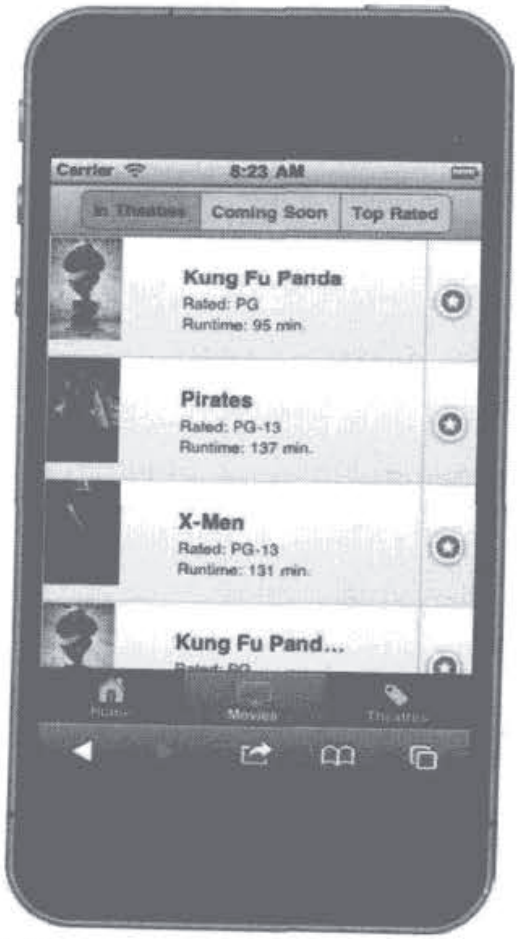


图 5-7 带有拆分列表的按钮

要创建一个拆分按钮，需要在列表条目内添加一个附属链接，jQuery Mobile 框架会添加一条垂直的线，以而分割主动作和附属动作（见程序清单 5-6）。

程序清单 5-6 带有拆分按钮的列表（ch5/list-split-buttons.html）

```
<ul data-role="listview" data-split-icon="star" data-split-theme="d">
    <li>
        <a href="movies/kung-fu-panda.html">
            <img src="images/kungfupanda2.jpg" />
            <h3>Kung Fu Panda</h3>
            <p>Rated: PG</p>
            <p>Runtime: 95 min.</p>
        </a>
        <a href="tickets.html">Buy Tickets</a>
    </li>
    ...
</ul>
```

要为所有的附属按钮设置图标，则需要为列表元素添加 data-split-icon 属性，并将其值设置为标准的（见表 4-1）或自定义的图标。默认情况下，附属按钮使用调色板颜色“b”（浅蓝色）来样式化。要应用其他主题，可以为列表元素添加 data-split-theme 属性。

5.6 编号列表

在使用有序列表（<ol>）时，需要创建编号列表（numbered list）（见图 5-8，相关代码见程序清单 5-7）。

程序清单 5-7 编号列表（ch5/list-unmbered.html）

```
<ol data-role="listview">
        <li><a href="spider-man.html">The Amazing Spider-Man</a></li>
        <li><a href="dark-knight.html">The Dark Knight Rises</a></li>
        ...
</ol>
```

默认情况下，jQuery Mobile 框架会在每一个列表条目的左边添加数字索引。在显示按顺序排列的一系列条目时，编号列表会很有用。例如，我们的“最受好评的（top rated）”电影视图（movie view）就可以使用一个编号列表，这是因为电影的排列顺序是与电影的受好评程度相关的。

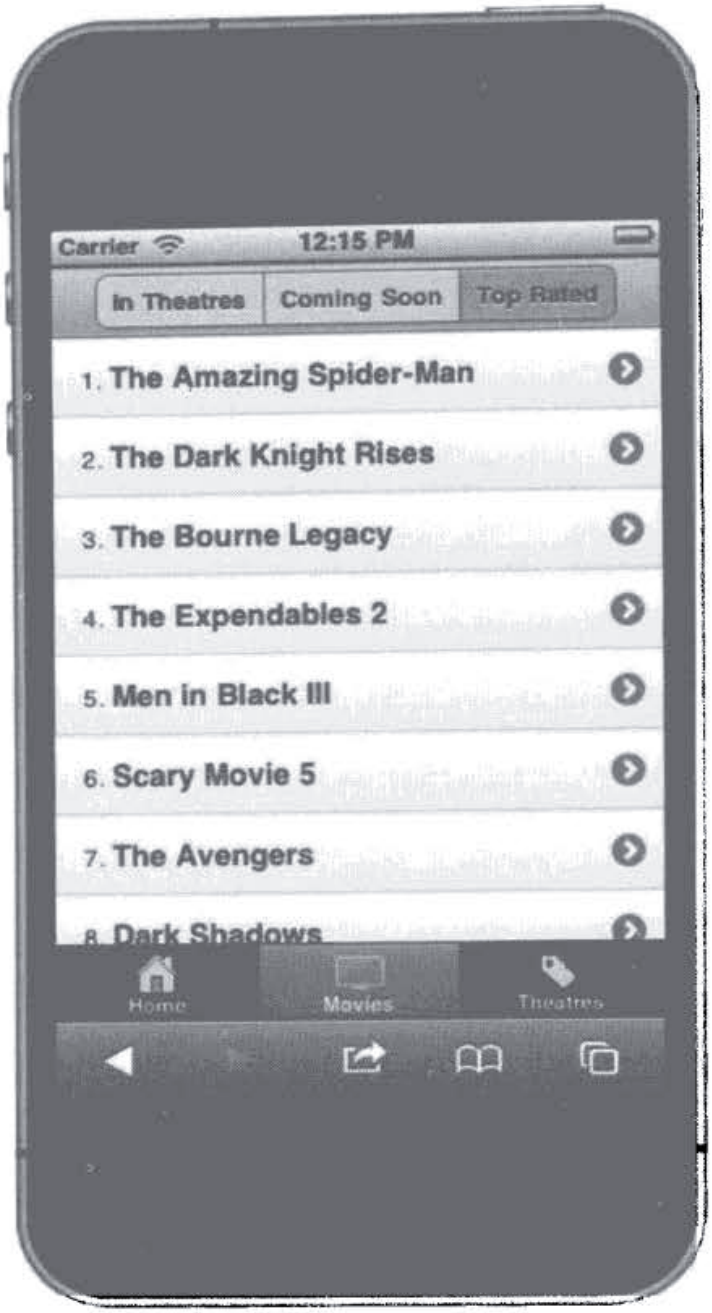


图 5–8 编号列表

5.7 只读列表

列表视图也可以显示只读的数据视图，而且用户界面看起来与前面出现的交互式界面非常相似，只不过纯图标的右箭头图像被移除，而且字体大小和内边距要略小一些。要创建一个只读的列表，只需移除前面例子中使用的锚标签即可（见图 5-9，相关代码见程序清单 5-8）。

图 5-9 带有只读条目的列表

程序清单 5-8 带有只读条目的列表（ch5/list-readonly.html）

```
<ul data-role="listview">
        <li>
                <img src="images/kungfupanda2.jpg" />
                <h3>Kung Fu Panda</h3>
                <p>Rated: PG</p>
                <p>Runtime: 95 min.</p>
        </li>
        ...
</ul>
```

5.8 列表徽章（计数泡）

列表徽章（list badge）或计数泡（count bubble）是一个突出显示的椭圆，通常用

来表示有多少个新的条目可供查看。例如，通常在邮件应用程序中使用的徽章用来指示用户有多少封未读邮件。在我们的例子中，徽章用来指示某个电影的评论是在几天前添加的（见图 5-10）。徽章可以用来表达任何类型的元数据。

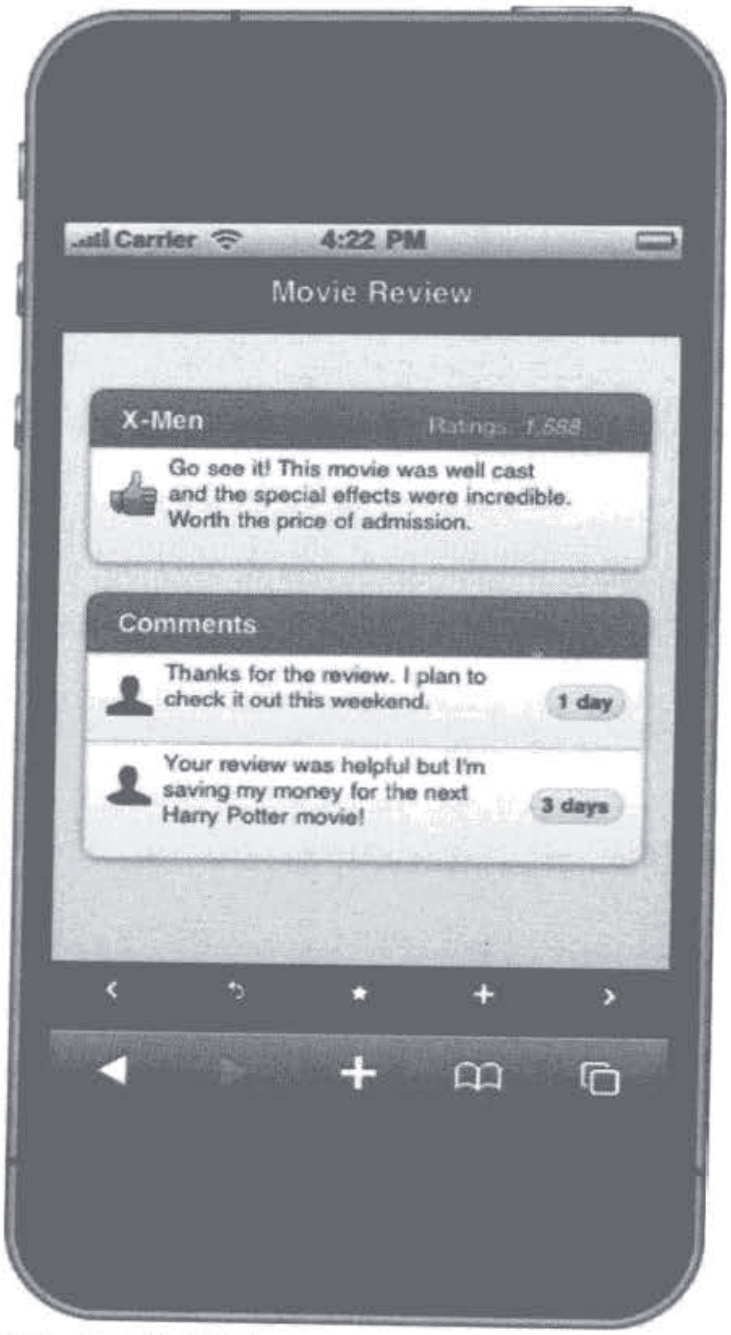


图 5-10 带有徽章或计数泡的列表

要创建一个徽章，需要使用一个包含 ui-li-count 类的元素对徽章中的文本进行包装。默认情况下，徽章使用调色板颜色“c”（灰色）来样式化。要应用其他主题，可以为列表元素添加 data-count-theme 属性（见程序清单 5-9）。

程序清单 5-9 带有徽章或计数泡的列表（ch5/list-badges.html）

```
<ul data-role="listview" data-inset="true" data-count-theme="e">
        <li data-role="list-divider">Comments</p></li>
        <li>
          <img src="../images/111-user.png" class="ui-li-icon">
          <p>Thanks for the review.  I'll check it out this weekend.</p>
          <span class="ui-li-count">1 day ago</span>
        </li>
</ul>
```

5.9 使用搜索栏过滤列表

jQuery Mobile 有一个非常方便的客户端搜索特性，可用于过滤列表。要创建一个搜索栏，需要为列表添加 data-filter="true"属性。jQuery Mobile 框架会在列表的上方添加一个搜索过滤器，而且其默认的占位符文本会显示“Filter items...”（见图 5-11，相关代码见程序清单 5-10）。

程序清单 5-10 列表过滤（ch5/list-filter.html）

```
<ul data-role="listview" data-filter="true" data-filter-
        placeholder="Search...">
    <li data-role="list-divider">
        Mon <p class="ui-li-aside">Feb 6 2012</p>
    </li>
    <li>
        <a href="b-day.html">
            <p>6 PM <span class="ui-li-aside">Birthday Party</span></p>
        </a>
    </li>
</ul>
```

有两种方法可以用于配置占位符文本。

1. 通过为列表元素添加 data-filter-placeholder 属性，可以配置占位符文本（见程序清单 5-10）。

2. 通过绑定 mobileinit 事件，并将 filterPlaceholder 选项设置为任何自定义的占位符的值，可以以全局方式将占位符的文本设置为 jQuery Mobile 的配置选项。

```
$(document).bind('mobileinit',function(){
 $.mobile.listview.prototype.options.filterPlaceholder="Search..";
});
```

第 8 章会详细讲解 jQuery Mobile 的配置。

当开始在搜索过滤器中输入文本时，客户端的过滤器会只显示与通配符搜索相匹

配的条目（见图 5-12）。

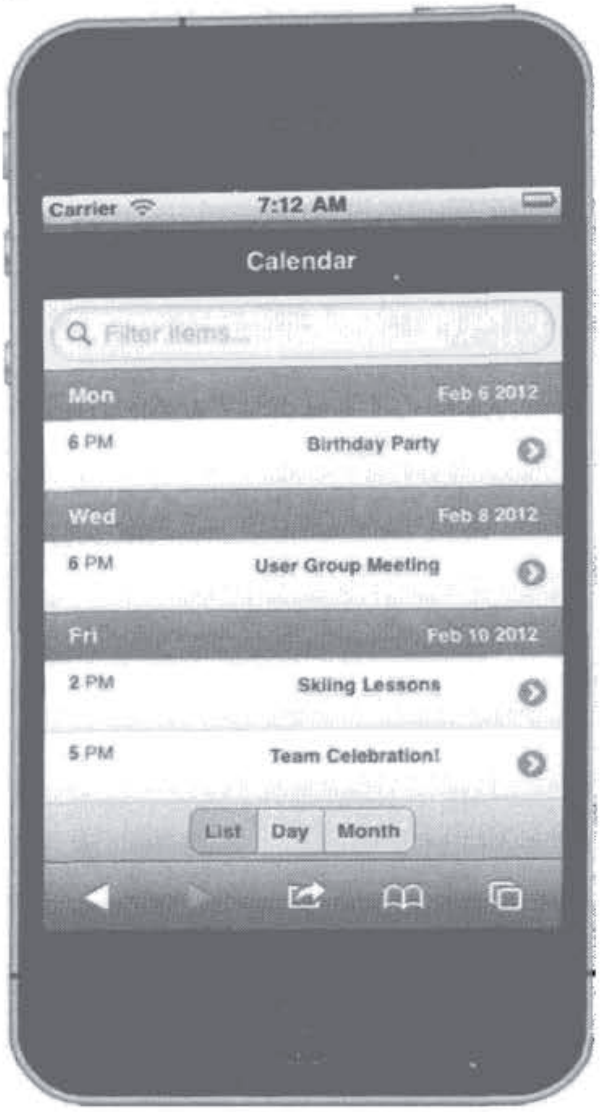


图 5-11 列表过滤（未过滤）

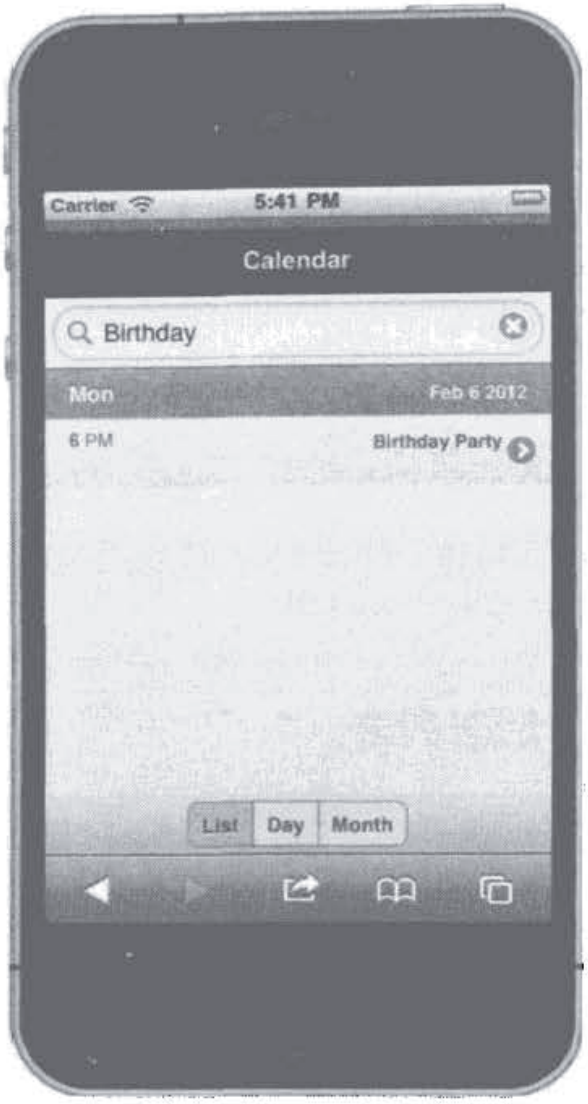


图 5-12 列表过滤（已过滤）

如果需要更改默认的搜索函数，有两种方法可以重写用于过滤的 callback（回调）函数。

第一种方法是，通过绑定 mobileinit 事件，并将 filterCallback 选项设置为任何自定义的搜索函数，从而以全局方式将搜索函数更新为 jQuery Mobile 的配置选项。例如，我们在这里将 callback 函数进行设置，使其使用一个“starts with”搜索。

```
$(document).bind('mobileinit',function(){
    $.mobile.listview.prototype.options.filterCallback =
      function( text, searchValue ){
          // Use a "starts with" search
          return !(text.toLowerCase().indexOf( searchValue ) === 0);
      };
});
```

callback 函数提供了两个参数：text 和 searchValue。text 参数包含列表条目的文本，而

searchValue 参数包含搜索过滤器的值。用于通配符搜索的默认行为以如下方式进行编码实现：

```
return text.toLowerCase().indexOf( searchValue ) === -1
```

如果 callback 函数针对某个列表条目返回了一个真（truthy）值，则该列表条目不会在搜索结果中显示（即该列表条目与搜索的内容不匹配）。

第二种方法是，在创建列表之后，我们可以动态的配置搜索函数。例如，在页面载入之后，我们可以为某个特定的列表应用新的搜索行为：

```
$("#calendar-list").listview('option', 'filterCallback',
        function( text, searchValue ) {
            // Use a "starts with" search
            return !(text.toLowerCase().indexOf( searchValue ) === 0);
        }
);
```

默认情况下，搜索框会继承其父容器的主题。要配置其他主题，则需要为列表元素添加 data-filter-theme 属性。

5.10 动态列表

listview 插件是一个能自动增强列表的微件。我们可以使用该插件来动态创建、更新列表。有两种方法可以用来创建动态列表：通过标记驱动的方法动态创建列表；通过显式设置 listview 插件的选项来动态创建列表（见程序清单 5-11）。

程序清单 5-11 listview 插件示例（ch5/dynmic-lists.html）

```
// Create list with markup-driven options
$( '<ul data-inset="true" id="list1">
        <li data-role="list-divider">Genres</li>
        <li><a href="#">Action</a></li>
        <li><a href="#">Comedy</a></li></ul>' )
   .insertAfter( "#list0" )
   .listview();

// Create list with plugin-driven options
$( '<ul><li data-role="list-divider">Genres</li>
        <li><a href="#">Action</a></li>
        <li><a href="#">Comedy</a></li></ul>' )
   .insertAfter( "#list1" )
   .listview({
        theme: "d",
        dividerTheme: "a",
        inset: true,
   });

// Add a new item to an existing list
$( "#list1" )
        .append('<li><a href="#">Drama</a></li>')
        .listview("refresh");
```

5.10.1 列表选项

listview 插件具有如下选项。

countTheme *string*
default: "c"

为徽章或计数泡设置主题调色板配色方案。这是一个取值范围为 a~z 的字母，它映射到你的主题中所包含的调色板。该选项还可以用作一个数据属性：data-count-theme= “a”。

```
$( “#list1” ).listview({ countTheme: "a" });
```

dividerTheme *string*
default: "b"

设置列表分割线的主题调色板配色方案。这是一个取值范围为 a~z 的字母，它映射到你的主题中所包含的调色板。该选项也可以用作一个数据属性：data-divider-theme= “a”。

```
$( “#list1” ).listview({ dividerTheme: "a" });
```

initSelector CSS selector string
default: ":jqmData(role='listview')"

initSelector 用来定义用于触发 widget 插件自动初始化的选择器（元素类型、数据角色[data role]等）。例如，由默认选择器匹配的所有元素都会被 listview 插件增强。为了重写该选择器，需要绑定 mobileinit 事件，并根据情况更新选择器。

```
$( document ).bind( "mobileinit", function(){
        $.mobile.listview.prototype.options.initSelector = "...";
        });
```

inset *boolean*
default: false

在该选项设置为 true 时，会创建一个内置列表。默认情况下（即该选项为 false），会创建一个基本列表。该选项也可以用作一个数据属性：data-inset= “true”。

```
$( “#list1” ).listview({ inset: true });
```

splitIcon *string*
default: "arrow-r"

在构建一个拆分按钮列表时，为附属按钮设置图标。该选项也可以用作一个数据属性：data-split-icon=“star”。

```
$( "#list1" ).listview({ splitIcon: "star" });
```

splitTheme *string*
default: "b"

在创建一个拆分按钮列表时，为附属按钮设置主题调色板配色方案。这是一个取值范围为 a~z 的字母，它映射到你的主题中所包含的调色板。该选项也可以用作一个数据属性：data-split-theme="a"。

```
$( "#list1" ).listview({ splitTheme: "a" });
```

theme *string*
default: "c"

设置列表的主题调色板配色方案。这是一个取值范围为 a~z 的字母，它映射到你的主题中所包含的调色板。该选项也可以用作一个数据属性：data-theme="a"。

```
$( "#list1" ).listview({ theme: "a" });
```

5.10.2 列表方法

listview 插件具有如下方法。

refresh：更新自定义列表。

该方法会更新自定义列表，以反映本地列表元素的值。例如，如果为一个现有的列表添加一个新的条目，我们必须调用“refresh”方法来重新构建列表条目。

```
// Add an item to an existing list and refresh the list item
$( "#list1" )
    .append('<li><a href="#">Drama</a></li>')
    .listview("refresh");

// Add list items to a new list and refresh the entire list
var markup = '<li>item 1</li><li>item 2</li>';
$( "#list2" )
    .append(markup)
    .listview( "refresh", true );
```

5.10.3 列表事件

listview 插件具有如下事件。

create：在创建列表时触发。

在创建一个自定义列表时，会触发该事件。它并不是用来创建一个自定义列表

```
$( '<ul><li data-role="list-divider">Genres</li>
    <li><a href="#">Comedy</a></li></ul>' )
  .insertAfter( "#list1" )
  .listview({
        inset: true,
        create: function(event) {
                console.log( "Creating list..." );
        }
  });
```

5.11 总结

在本章中，我们讲解了广受欢迎的列表视图控件。列表视图之所以能够被广泛使用，是因为通过它们，用户能够简单而且有效地进行浏览体验。jQuery Mobile 列表能够以多种独特的方式进行样式化和配置。无论是基本列表，还是带有图像、拆分按钮、分割线或勋章的列表，我们都有多个配置选项进行选择。

我们还讲解了如何将一个搜索过滤器简单地添加到列表中，以及如何在必要时重写默认的搜索函数的示例。

最后，我们讲解了 listview 插件 API，以及如何动态创建和更新列表，从而为用户提供更多的互动体验的示例。

在下一章，我们会讲解另外一个广受欢迎的用户界面组件：jQuery Mobile 灵活的表格布局。我们会学到如何使用表格来创建响应式设计，以及如何使用 CSS 渐变来增强我们的用户界面。

第6章

使用表格和CSS渐变来格式化内容

当内容需要以一种灵活的方式编组到区域（section）时，移动应用程序通常会为之应用表格。在需要该行为的设计中，jQuery Mobile 的自适应表格就是一种有效的解决方案。在本章中，我们会讲解 jQuery Mobile 表格组件的基本知识，并通过几个示例来讲解如何在表格中样式化图标、图形和文本。我们还会创建可折叠的内容块，并讨论它与内嵌的（inline）页面结构相比较时，所具有的优势。我们会使用 CSS 渐变来修饰一下我们的设计，并讨论 CSS 渐变在性能和渐进式增强方面所提供的优势。

6.1 表格布局

jQuery Mobile 的表格是可配置的，它可以支持 2～5 列的表格布局。从 HTML 的角度来看，表格是使用 CSS 属性配置的 div 元素。表格相当灵活，而且会占据显示屏幕的整个宽度。表格不包含边界、内间距（padding）、边距（margin），这样就不会对其内部包含的元素的样式形成干扰。在我们查看示例之前，首先来讲解标准的表格模板。

6.1.1 表格模板

在创建多列表格时，表格模板将是一个非常有用的参考（见程序清单 6-1）。

程序清单 6-1 表格模板

```
<div data-role="content">

  <!-- Grid container -->
  <div class="<grid-css-attribute>">
    <!-- Blocks -->
    <div class="<block-css-attribute>">Block A</div>
    <div class="<block-css-attribute>">Block B</div>

  </div>
</div>
```

在创建表格时，需要创建具有两个或更多个内层块（inner block）的外层表格容器。

1. 表格容器（grid container）：表格容器需要 CSS 属性 ui-grid-*来配置表格中列的数量（见表 6-1）。例如，要创建一个两列的表格，我们需要将表格 CSS 属性设置为 ui-grid-a。

表 6-1 表格 CSS 属性参考

列的数量	表格 CSS 属性
2	ui-grid-a
3	ui-grid-b
4	ui-grid-c
5	ui-grid-d

2. 块（block）：块包含在表格内。块需要 CSS 属性 ui-block-*来识别其列的位置（见表 6-2）。例如，如果我们有一个两列的表格，则第一个块会用 CSS 属性 ui-block-a 来样式化，而第二个块会用 CSS 属性 ui-block-b 来样式化。

表 6-2 块 CSS 属性参考

第 X 列	块 CSS 属性
1	ui-block-a
2	ui-block-b
3	ui-block-c
4	ui-block-d
5	ui-block-e

6.1.2 两列的表格

一个两列的表格（50%，50%）示例如图 6-1 所示，其相关代码见程序清单 6-2。

图 6–1 两列的表格

程序清单 6-2 两列的表格（ch6/grid-2col.html）

```
<div data-role="content">
  <div class="ui-grid-a">
    <div class="ui-block-a">Block A</div>
    <div class="ui-block-b">Block B</div>
  </div>
</div>
```

外层表格（outer grid）使用 CSS 表格属性 ui-grid-a 进行配置。接下来，我们添加两个内层块。第一个块被分配了一个 CSS 属性 ui-block-a，第二个块被分配了一个 CSS 属性 ui-block-b。如图 6-1 所示，列是等间距、无边界的，而且每个块内的文本在必要时会换行显示。作为一个额外的优点，jQuery Mobile 内的表格相当灵活，而且会根据不同的屏幕显示尺寸以自适应的方式进行呈现（见图 6-2）。

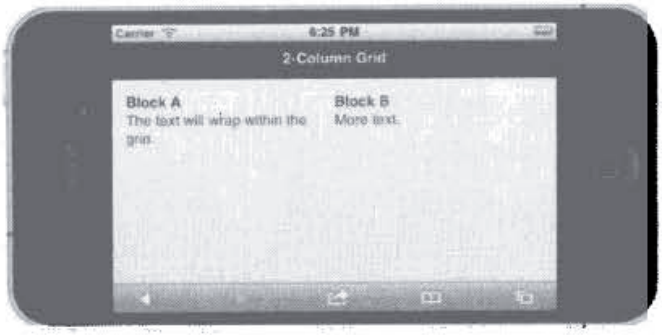


图 6–2 两列的表格（横屏模式）

6.1.3 带有CSS增强的三列表格

一个三列表格（33%，33%，33%）的示例如图6-3所示，其相关代码见程序清单6-3。

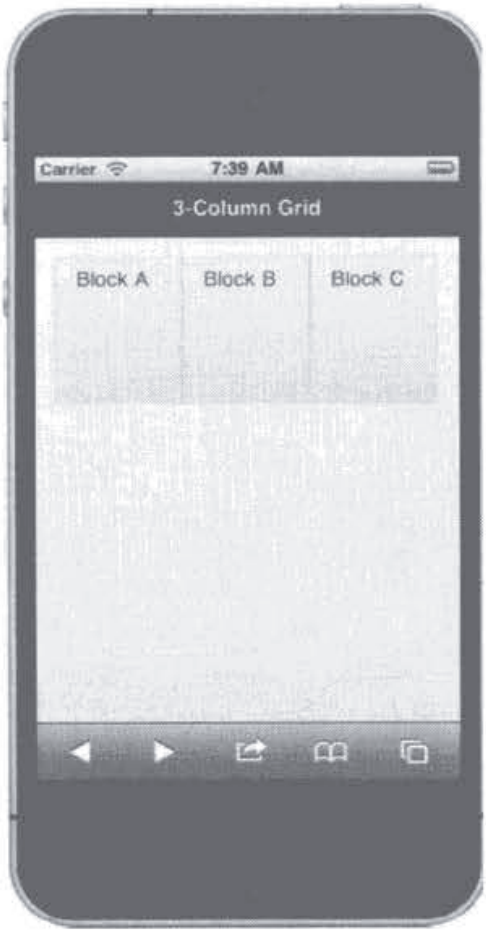


图6-3 带有CSS增强的三列表格

程序清单6-3 三列表格（ch6/grid-3col.html）

```
<div data-role="content">
    <div class="ui-grid-b">
       <div class="ui-block-a">
          <div class="ui-bar ui-bar-e" style="height:100px">Block A</div>
       </div>
       <div class="ui-block-b">
          <div class="ui-bar ui-bar-e" style="height:100px">Block B</div>
       </div>
       <div class="ui-block-c">
          <div class="ui-bar ui-bar-e" style="height:100px">Block C</div>
       </div>
    </div>
</div>
```

它与前面看到的两列表格非常相似，但是它配置了 CSS 属性（ui-grid-b）以支持三个列，而且我们为第三列（ui-block-c）添加了一个额外的块。我们还使用可主题化的类（可以添加到包含了表格的任何元素上）对块进行了样式化。在本例中，我们添加了 ui-bar 以应用 CSS 内间距，还添加了 ui-bar-e 以便为“e”工具栏主题调色板应用背景渐变和字体样式。你可以使用范围为 a～e 内的任何工具栏主题（ui-bar-*）来样式化你的块。最后，为了创建一致的块高度，我们还对高度以内嵌方式进行了样式化（style=“height:100px”）。从视觉上看，这些增强都是使用线性的背景渐变来样式化我们的表格，现在块与块之间使用边界进行隔离。

6.1.4 带有 app 图标的四列表格

一个四列表格（25%，25%，25%，25%）的示例如图 6-4 所示，其相关代码见程序清单 6-4。

图 6-4 带有 app 图标的四列表格

程序清单 6-4 四列表格（ch6/grid-4col.html）

```
<div data-role="content">
  <div class="ui-grid-c" style="text-align: center;">
    <div class="ui-block-a"><img src="images/cloud-default.png"></div>
    <div class="ui-block-b"><img src="images/cloud-bright.png"></div>
    <div class="ui-block-c"><img src="images/cloud-ripple.png"></div>
    <div class="ui-block-d"><img src="images/cloud-sparkle.png"></div>
  </div>
</div>
```

它与三列表格相似，只不过为该表格配置了 CSS 属性（ui-grid-c），以支持 4 个列，而且我们为第四列（ui-block-d）添加了一个额外的块。此外，出于平衡和一致性考虑，我们将 app 图标放置在表格的中央位置（style= “text-align: center;”）。从视觉上看，这个表格具有个大小相等的 app 图标，它与应用程序 springboard 非常相似。

6.1.5 带有 Emoji 图标的五列表格

一个五列表格（20%，20%，20%，20%，20%）的示例如图 6-5 所示，其相关代码见程序清单 6-5。

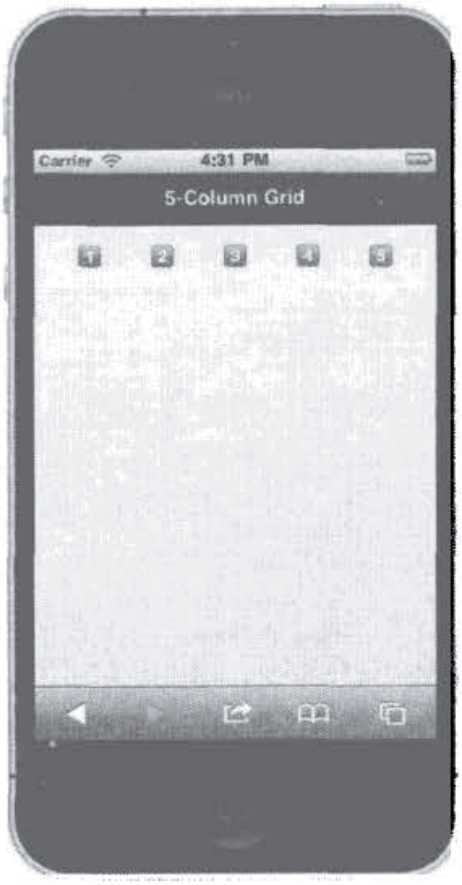


图 6-5 五列表格

程序清单 6-5 五列表格（ch6/grid-5col.html）

```
<div data-role="content">
  <div class="ui-grid-d" style="text-align: center;">
    <div class="ui-block-a">&#xe21c;</div>
    <div class="ui-block-b">&#xe21d;</div>
    <div class="ui-block-c">&#xe21e;</div>
    <div class="ui-block-d">&#xe21f;</div>
    <div class="ui-block-e">&#xe220;</div>
  </div>
</div>
```

该表格与前面讲解的四列表格非常相似，只不过为该表格配置了 CSS 属性（ui-grid-d），以支持 5 个列，而且我们为第五列（ui-block-e）添加了一个额外的块。每一个块都包含独特的 Emoji 图标[1]。

> **注意：** Emoji 图标能够替代图像，而且它无需 HTTP 请求，其负载只有少量的文本字符，因此性能出色。不幸的是，Emoji 图标当前只能支持 iOS 系统。

6.1.6 多行表格

目前为止，我们看到的表格都只有一行。为了添加其他行，只需为其简单地重复第一行的块模式即可（见图 6-6，相关代码见程序清单 6-6）。我们最终生成的是一个三行五列的表格。列的宽度都是相等的，而且你在块组件上手动调整行的高度。

图 6-6 多行表格

[1] 见 http://pukupi.com/post/1964.

程序清单 6-6 多行表格（ch6/grid-multi-row.html）

```
<div data-role="content">
  <div class="ui-grid-d" style="text-align: center;">

    <!-- First row -->
    <div class="ui-block-a">&#xe21c;</div>
    <div class="ui-block-b">&#xe21d;</div>
    <div class="ui-block-c">&#xe21e;</div>
    <div class="ui-block-d">&#xe21f;</div>
    <div class="ui-block-e">&#xe220;</div>
    <!-- Second row -->
    <div class="ui-block-a">&#xe002;</div>
    <div class="ui-block-b">&#xe005;</div>
    <div class="ui-block-c">&#xe51a;</div>
    <div class="ui-block-d">&#xe515;</div>
    <div class="ui-block-e">&#xe152;</div>

  </div>
</div>
```

6.1.7 不规则的表格

目前为止，我们看到的每一个表格，列的宽度都是相等的，这是因为 jQuery Mobile 在默认情况下会平均地来划分所有的列。但是，如果需要自定义列的尺寸，可以在 CSS 中调整其宽度。例如，通过设置每一个块的自定义宽度，我们可以将两列表格的宽度修改为一个 25%，75%的表格（见图 6-7，相关代码见程序清单 6-7）。因此，我们的表格可以进行修改，以支持各种尺寸。

图 6-7 不规则的表格

程序清单 6-7 不规则的表格（ch6/grid-uneven.html）

```
<style>
        /* Set 2-column grid to 25/75% */
        .ui-grid-a .ui-block-a {
            width: 25%;
        }
        .ui-grid-a .ui-block-b {
            width: 75%;
        }

        /* Set 3-column grid to 25/50/25% */
        .ui-grid-b .ui-block-a {
            width: 25%;
        }
        .ui-grid-b .ui-block-b {
            width: 50%;
        }
        .ui-grid-b .ui-block-c {
            width: 25%;
        }
</style>
```

6.1.8 springboard

springboard 是一个可以应用我们的表格布局的理想物件。在下面的例子中，我们会看到两种类型的 springboard。首先，我们会看到使用 app 图标进行样式化的 springboard（见图 6-8），然后我们会看到使用 Glyphish 图标样式化的 springboard（见图 6-9）。

图 6-8 带有 app 图标的 springboard

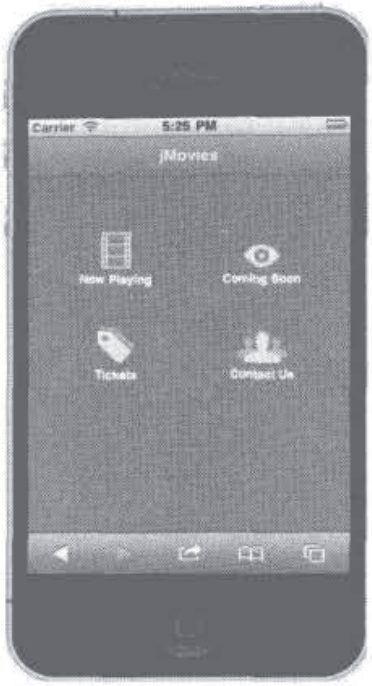


图 6–9 带有 Glyphish 图标的 springboard

准备好应对 springboard 的挑战了么？如果准备好了，先创建一个与上面任意一个图相似的 springboard 吧。从表格的角度来看，这两个示例的配置方式相同。但是，为了适应不相等的图标高度，带有 Glyphish 图标的 springboard（见程序清单 6-9）与带有 app 图标的 springboard（见程序清单 6-8）相比，在样式化的方式上略有不同。

程序清单 6-8 带有 app 图标的 springboard（ch6/springboard1.html）

```
<div class="ui-grid-a">
  <div class="ui-block-a">
    <a href="#">
      <img src="images/cloud.png">
      <span class="icon-label">App A</span>
    </a>
  </div>
  …
</div>

<style>
  /* center icons */
  .ui-grid-a { text-align: center; }

  /* set row height */
  .ui-block-a, .ui-block-b { height: 100px; }

  /* set label color and size */
  .icon-label { color: #000; display: block; font-size:12px; }

  a:link, a:visited, a:hover, a:active { text-decoration:none; }
</style>
```

程序清单 6-9 带 Glyphish 图标的 springboard（ch6/springboard2.html）

```
<div class="ui-grid-a">
  <div class="ui-block-a">
    <div class="icon-springboard">
      <a href="#">
        <img src="images/45-movie-1-lg.png" alt="Now Playing">
        <span class="icon-label">Now Playing</span>
      </a>
    </div>
  </div>
  …
</div>

<style>
  /* center icons */
  .ui-grid-a { text-align: center; }

  /* set row height */
  .ui-block-a, .ui-block-b { height: 100px; position: relative; }

  /* set label size and color */
  .icon-label { color: #FFF; display: block; font-size:12px; }

  /* bottom align icons to adjust for uneven icon heights */
  .icon-springboard { position: absolute; bottom: 0; width: 100%; }

  a:link, a:visited, a:hover, a:active { text-decoration:none; }
</style>
```

6.2 可折叠的内容块

你有没有发现，当你在阅读整个移动页面的内容时，需要反复滚动页面？尽管这可以锻炼到你的手指，但是由于用户必须反复滚动页面，因此用户体验相当糟糕。如果你正在查找某种替换方式，你可能会考虑将内容编组到可折叠的内容块中。

> **提示：**与内嵌的页面结构相比，可折叠的内容块具有很多优势。首先，我们可以将内容折叠到分段的组中，以让它们在单个视图中都是可见的（见图 6-10）。此外，由于我们淘汰了滚动操作，用户的体验也会提升。

用来创建可折叠内容块的标记如程序清单 6-10 所示。

程序清单 6-10 可折叠的内容块（ch6/collapsible-block.html）

```
<div data-role="content">

  <div data-role="collapsible" data-collapsed="true" data-theme="a" data-content-theme="b">
```

```
    <h3>Wireless</h3>
    <ul data-role="listview" data-inset="true">
      <li><a href="#">&#xe117; Notifications</a></li>
      <li><a href="#">&#xe01d; Location Services</a></li>
    </ul>
  </div>

  <div data-role="collapsible" data-theme="a" data-content-theme="b">
    <h3>Applications</h3>
    <ul data-role="listview" data-inset="true">
      <li><a href="#">&#xe001; Faceoff</a></li>
      <li><a href="#">&#xe428; LinkedOut</a></li>
      <li><a href="#">&#xe03d; Netflicks</a></li>
    </ul>
  </div>

</div>
```

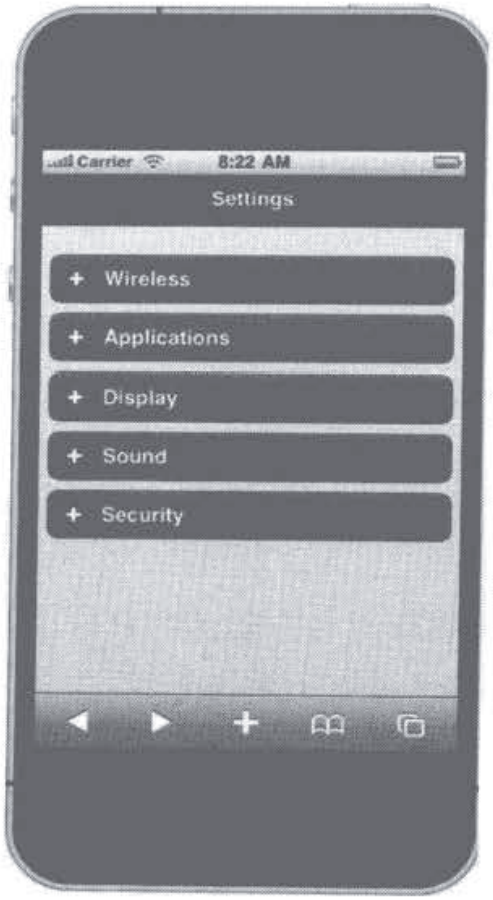


图 6-10 内容块（所有块为折叠状态）

在创建可折叠的内容块时，需要两个元素。

1. 创建一个容器并添加 data-role="collapsible" 属性。你也可以通过添加 data-collapsed 属性将容器配置为折叠的或展开的。默认情况下，可折叠的区域块将会以展开方式显示（data-collpased="false"）。为了在最初以折叠方式显示区域块，需要为容

器添加 data-collpased="true" 属性。例如，我们启动程序清单 6-10 中的代码，则最初的视图如图 6-11 所示。在程序清单中，除了默认情况下为展开状态的“Application”区域块之外，其他所有的内容块都已经显式设置为折叠状态。

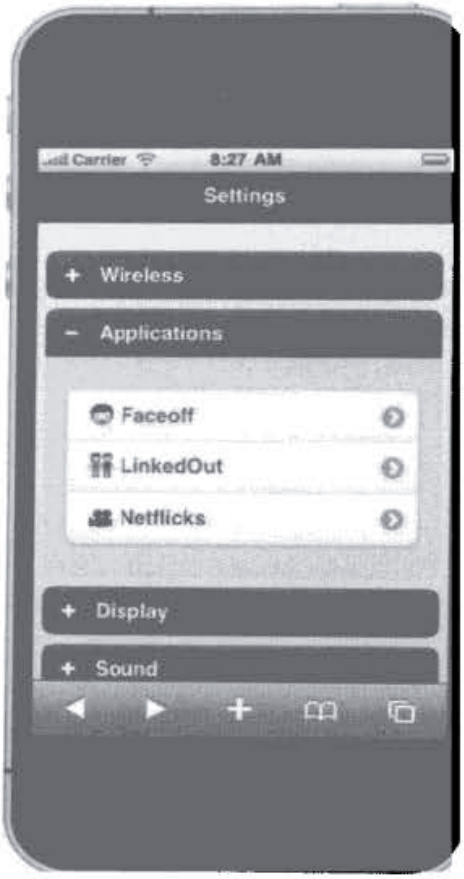


图 6-11 内容块（一个区域块为展开状态）

2．在容器内，添加任意的页眉元素（H1～H6）。jQuery Mobile 框架会对页眉进行样式化，使其看起来就像是一个带有左对齐的加号图标或减号图标的可单击按钮，其中加号图标或减号图标用来指示该容器是否是展开的。

在页眉之后，可以为可折叠的区域块添加任何 HTML 标记。jQuery Mobile 框架会将该标记包含在容器内，当用户轻敲页眉时，该容器或者是展开，或者是折叠。通过为可折叠的容器添加 data-theme 和 data-content-theme 属性，可以分别主题化可折叠的块和与其相关联的按钮（见程序清单 6-10）。

> **注意**：可折叠的块允许用户同时让多个块呈展开状态或折叠状态（见图 6-12）。在下一节，我们在处理可折叠的设置时，情况就不是这样了。

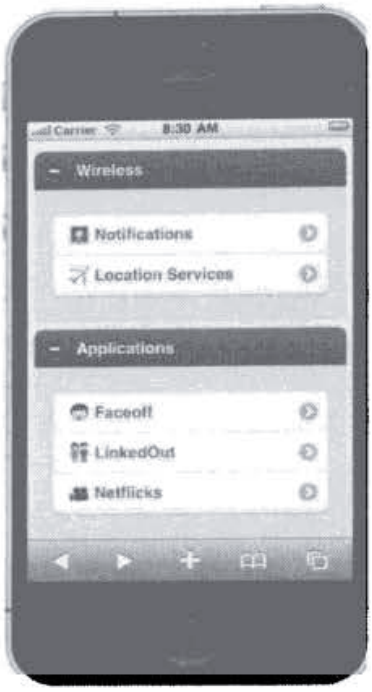


图 6–12 内容块（所有的块都展开）

6.3 可折叠的设置

可折叠的设置（见图 6-13）与可折叠的块相似，只不过它的可折叠的区域在视觉上是组合在一起的，而且一次只能展开一个区域，这使得可折叠的设置的外观就像手风琴那样（见图 6-14）。

图 6–13 内容设置（折叠状态）

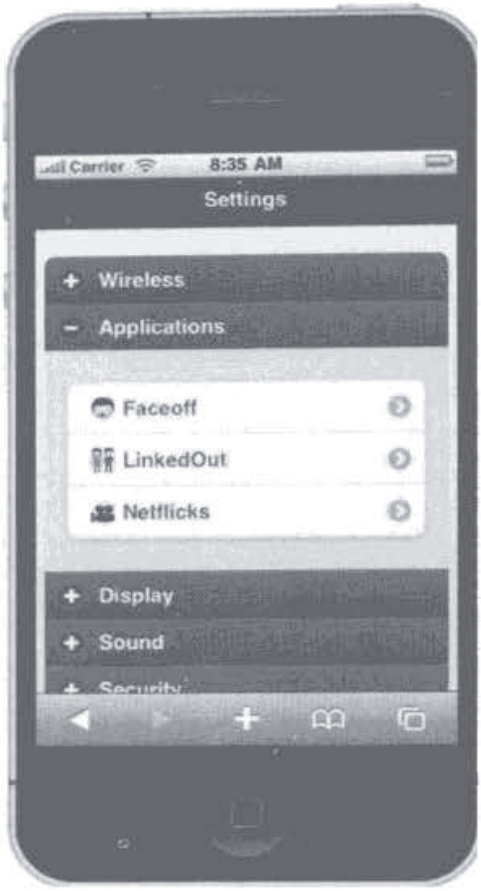


图 6-14 内容设置（展开状态）

在设置内打开一个新的区域时，之前展开的任何区域都会自动折叠起来。

用于可折叠设置的标记与构建可折叠块时使用的标记相同。然而，为了创建手风琴样式的行为和编组，我们需要使用 data-role="collapsible-set"添加一个父包装（parent wrapper），如程序清单 6-11 所示。通过为可折叠的设置添加 data-theme 和 data-content-theme 属性，可以分别主题化可折叠的区域和与其相关联的按钮。

程序清单 6-11 可折叠的设置（ch6/collapsible-set.html）

```
<div data-role="content">

  <div data-role="collapsible-set" data-theme="a" data-content-theme="b">
    <div data-role="collapsible" data-collapsed="true">
      <h3>Wireless</h3>
      <ul data-role="listview" data-inset="true">
        <li><a href="#">&#xe117; Notifications</a></li>
        <li><a href="#">&#xe01d; Location Services</a></li>
      </ul>
    </div>

    <div data-role="collapsible">
      <h3>Applications</h3>
```

```
        <ul data-role="listview" data-inset="true">
          <li><a href="#">&#xe001; Faceoff</a></li>
          <li><a href="#">&#xe428; LinkedOut</a></li>
          <li><a href="#">&#xe03d; Netflicks</a></li>
        </ul>
      </div>
      ...
    </div><!-- /collapsible-set -->
  </div>
```

6.4 使用CSS渐变进行样式化

打算装饰一下你的移动UI？可以在你通常使用背景图像的地方使用CSS渐变。CSS渐变能够替代图片，而且性能更好，它能够很好地适用于灵活的布局，而且当浏览器不提供支持时，也可以优雅地降级。例如，通过添加渐变，我们可以将一个原始的springboard（见图6-15）以一种更为优雅的方式显示出来（见图6-16和图6-17）。

图6-15 没有应用CSS渐变的springboard

图 6–16 应用了 CSS 渐变的 springboard（iOS）

图 6–17 应用了 CSS 渐变的 springboard（Android）

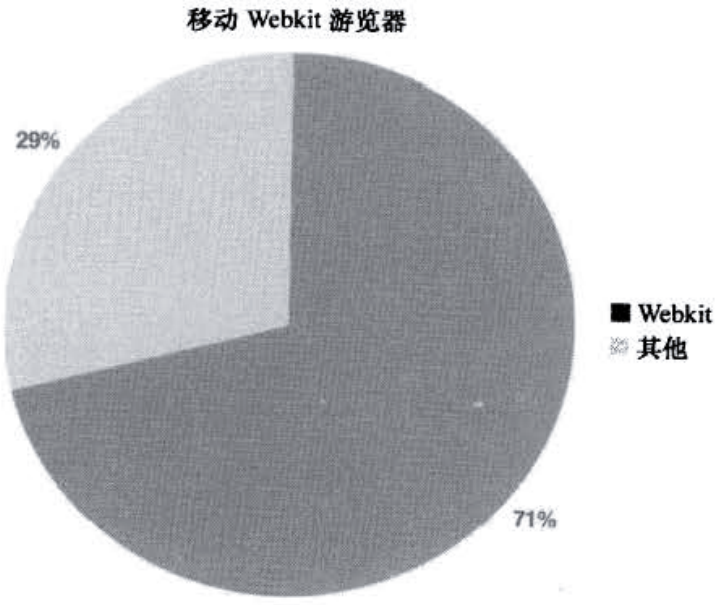
但凡使用背景图像的地方，就可以使用渐变。例如，渐变通常用于样式化你的页眉、内容和按钮的背景。此外，有两种类型的CSS渐变：线性渐变和放射性渐变。线性渐变相对来说比较简单，如果你不熟悉它的语法，则可以使用CSS渐变生成器[1]来帮助你起步。生成背景线性渐变的CSS如程序清单6-12所示。

程序清单6-12 背景渐变

```
.background-gradient {
  background-image: -webkit-gradient(
    linear, left bottom, left top,
    color-stop(0.22, rgb(92,92,92)),
    color-stop(0.57, rgb(158,153,158)),
    color-stop(0.84, rgb(92,92,92))
  );
}

<!-- Set the gradient on the page -->
<div data-role="page" class="background-gradient">
```

尽管这一CSS渐变针对的是最流行的WebKit布局引擎（见图6-18），通过包含其厂商特定的前缀，你也可以添加对其他浏览器的支持。



http://en.wikipedia.org/wiki/Mobile_browser#Popular_mobile_browsers

图6-18 WebKit的使用

例如，为了在Mozilla浏览器上呈现渐变，我们需要添加-moz-厂商前缀版本（见程序清单6-13）。

[1] 见http://www.westciv.com/tools/gradients/或http://gradients.glrzad.com/.

程序清单 6-13 支持 Mozilla 浏览器的背景渐变

```
.background-gradient {
  background-image: -webkit-gradient(
    linear, left bottom, left top,
    color-stop(0.22, rgb(92,92,92)),
    color-stop(0.57, rgb(158,153,158)),
    color-stop(0.84, rgb(92,92,92))
  );
  background-image: -moz-linear-gradient(
    90deg,
    rgb(92,92,92),
    rgb(158,153,158),
    rgb(92,92,92));
}
```

用于页眉的渐变实际上是三个独立渐变的叠加，其中包含一个线性渐变和两个放射性渐变。放射性渐变会创建一个圆形的渐变效果。用来创建页眉渐变的代码如程序清单 6-14 所示。

程序清单 6-14 springboard 渐变

```
.header-gradient {
  background-image:
    -webkit-gradient(
      linear, left top, left bottom,
      from( rgba( 068,213,254,0 )),
      color-stop(.43, rgba( 068,213,254,0 )),
      to( rgba( 068,213,254,1 ))),
    -webkit-gradient( radial,
      50% 700, 690,
      50% 700, 689,
      from( rgba( 049,123,220,0 )),
      to( rgba( 049,123,220,1 ))),
    -webkit-gradient(
      radial,
      20 -43, 60,
      20 -43, 40,
      from( rgba( 125,170,231,1 )),
      to( rgba( 230,238,250,1 )));
}

<!-- Set the gradient on the header -->
<div data-role="header" class="header-gradient">
```

6.5 总结

在本章，我们讲解了 jQuery Mobile 基于表格的设计的用途，以及如何迅速地样式化表格模板内的内容（见程序清单 6-1）。当内容需要以一种灵活的方式编组到区域

（section）时，jQuery Mobile 的表格就是一种理想的解决方案。我们的表格可以包含任何内容，我们还查看了几个使用文本、图标和图形来样式化的表格示例。

我们还讲解了可折叠的内容块，并讨论了它们与内嵌的页面结构相比较时所具有的优势。可折叠的块是一种有效的可用模式，它们可以在单个视图中显示所有内容，而且将滚动页面从用户体验中剔除。因此，用户使用应用程序时，其体验也得以提升。

最后，我们讲解了如何使用 CSS 渐变来装饰我们的设计。CSS 渐变能够替代图片，而且性能更好，它能够很好地适用于灵活的布局，而且当浏览器不提供支持时，也可以优雅地降级。

在第 7 章，我们会继续讲解布局设计，并仔细讲解 jQuery Mobile 的主题化框架。

第 7 章

创建可主题化的设计

jQuery Mobile 有一个内置的主题框架，它允许设计人员迅速地自定义和重新样式化他们的用户界面。该主题框架使用了许多 CSS3 的特性，后者有助于构建更为优雅和响应式的设计。例如，通过使用 CSS3，主题框架能够在不使用图片的情况下，实现圆角、阴影和渐变功能。这是一种性能优势，因为框架在无需造成额外 HTTP 请求的情况下，就可以提供更具吸引力的界面。实质上，我们拥有的是一个能够在所有浏览器上呈现出统一设计的轻量级主题框架。

在本章中，我们会讲解主题框架的基础知识，以及 jQuery Mobile 包含的默认主题。我们还会讨论可以为组件分配主题的三种方式。尽管所有的组件都可以通过 data-theme 属性来显式地设置其主题，但是大多数组件都有默认的主题，并且也能继承其父容器的主题。我们会讨论每一种方式的优势和应用实例，以及为组件分配主题时，需要考虑的优先级情况。

最后，我们会讲解如何创建自定义的主题。如果需要创建更为丰富的设计，或者是需要创建一个最能匹配公司品牌文化的设计时，就有必要自定义主题了。有两个方式可用来创建自定义主题，我们会详细讲解每一种方式的示例。第一种方式是手动方式，这可以让设计人员完全控制其布局；而第二种方式是使用 ThemeRoller[1]，这是一种基于 Web 的工具，能够自动创建新的主题。

---

1 见 http://jquerymobile.com/themeroller.

7.1 主题基础知识

在多个例子中，我们已经学习了如何使用 data-theme 属性为页面容器（页面、页眉、内容、页脚）和表单元素应用其他主题。例如，我们可以使用一个未主题化的页面（见图 7-1），然后使用不同的页眉和列表主题（添加了简单的 data-theme 属性，见程序清单 7-1）对其重新样式化（见图 7-2）。

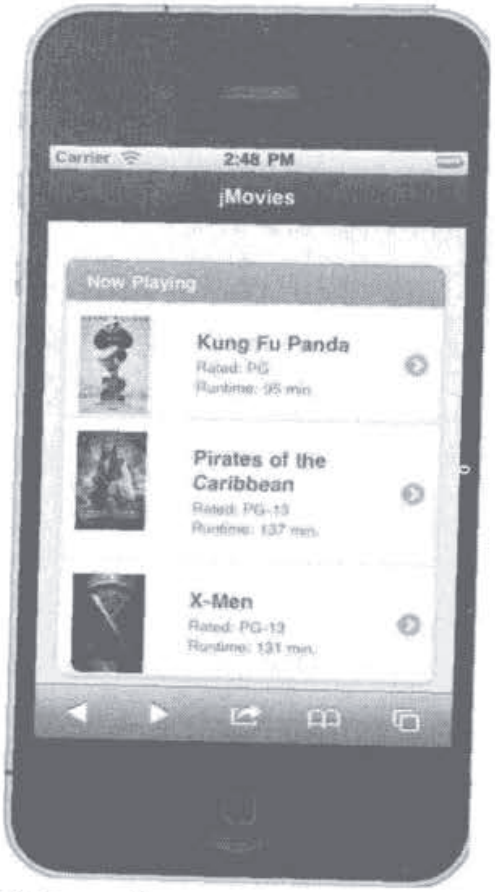


图 7-1 带有默认主题的列表

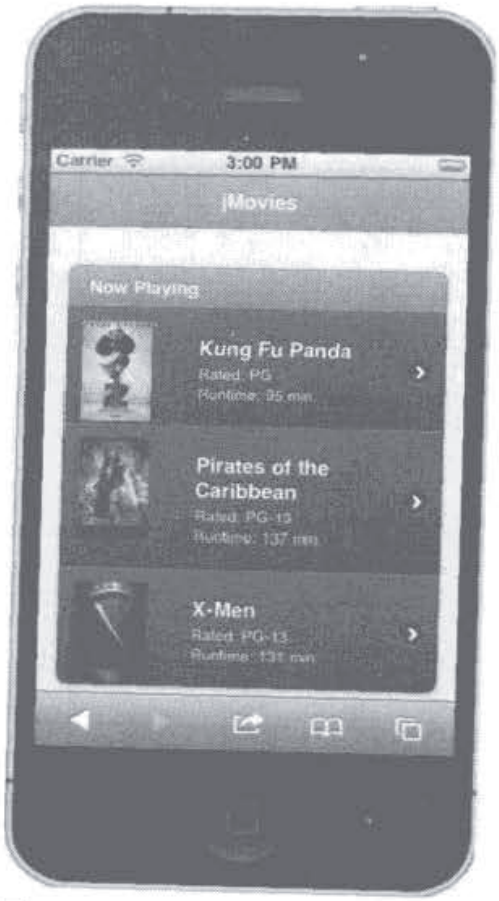


图 7-2 带有其他主题的列表

程序清单 7-1 data-theme 属性（ch7/theme-list 2.html）

```
<div data-role="page">
        <div data-role="header" data-theme="b">
                <h1>jMovies</h1>
        </div>

        <div data-role="content">
                <ul data-role="listview" data-inset="true" data-theme="a">
                        <li data-role="list-divider">Now Playing</li>
                </ul>
        </div>
</div>
```

7.2 主题和调色板

jQuery Mobile CSS 文件总是我们最先导入到页眉元素中的资源（asset）（见程序清单 7-2）。该文件包含用于 jQuery Mobile 应用程序的默认结构和主题。先花些时间使用你最喜欢的编辑器来研究一下该文件的内容。

程序清单 7-2 导入的 jQuery Mobile CSS 文件

```
<head>
   <link rel="stylesheet" type="text/css" href="jquery.mobile-min.css" />
   <script type="text/javascript" src="jquery-min.js"></script>
   <script type="text/javascript" src="jquery.mobile-min.js"></script>
</head>
```

jQuery Mobile CSS 文档包含两个部分：主题部分和结构部分。

- 文档前半部分包含默认的主题设置。主题设置管理所有组件的可视化样式（背景、边界、颜色、字体和阴影）。在设置 data-theme 主题时，我们具有 5 个不同的可选项（a、b、c、d、e）。从技术角度上，这些字母（a～z）被称为调色板。在查看 jQuery Mobile CSS 文件时，你可能会注意到，CSS 文件内出现的第一个调色板是调色板“a”（见程序清单 7-3）。

程序清单 7-3 jQuery Mobile CSS 调色板“a”（部分清单）

```
/* A
------------------------------------------------------------------------*/
.ui-bar-a {
        border: 1px solid                #2A2A2A;
        background:                      #111111;
        color:                                  #ffffff;
        font-weight: bold;
        text-shadow: 0 -1px 1px #000000;
        ...
        background-image:          linear-gradient(top, #3c3c3c, #111);
}
.ui-body-a {
        border: 1px solid                #2A2A2A;
        background:                      #222222;
        color:                  #fff;
        text-shadow: 0 1px 0    #000;
        font-weight: normal;
        background-image:          linear-gradient(top, #666, #222);
}
...
```

主题部分进一步细分为如下子部分。

■ 调色板（swatch）：默认情况下，jQuery Mobile 有 5 个调色板可供选择（a、b、c、d、e），你可以根据需要添加多个独特的调色板。调色板允许我们为所有的组件配置独特的背景、边界、颜色、字体和阴影。方便起见，用于新调色板的命名约定是基于字母的（a～z）。但是，调色板名字的长度没有任何限制。在本章后面，我们会看到创建自定义调色板的几个示例。

■ 全局主题设置（global theme settings）：全局主题设置时在调色板之后配置的。这些设置为按钮添加了视觉上的样式增强，比如圆角、图标、叠加（overlay）和阴影。由于这些设置是全局的，因此会被所有的调色板配置继承（见程序清单 7-4）。

程序清单 7-4 jQuery Mobile 全局主题设置（部分清单）

```
/* Active class used as the "on" state across all themes
------------------------------------------------------------------------*/
.ui-btn-active {
        border: 1px solid                #155678;
        background:                      #4596ce;
        font-weight: bold;
        color:                  #fff;
        cursor: pointer;
        text-shadow: 0 -1px 1px #145072;
        text-decoration: none;
        ...
}
```

■ 结构（structure）：jQuery Mobile CSS 文件的后半部分包含结构样式，其中主要包含定位、内间距、边距、高度和宽度设置（见程序清单 7-5）。

程序清单 7-5 jQuery Mobile 结构样式（部分清单）

```
/* some unsets - more probably needed */
.ui-mobile, .ui-mobile body { height: 100%; }
.ui-mobile fieldset, .ui-page { padding: 0; margin: 0; }
.ui-mobile a img, .ui-mobile fieldset { border: 0; }

...

.ui-checkbox, .ui-radio {
        position:relative;  margin: .2em 0 .5em; z-index: 1;
}
.ui-checkbox .ui-btn, .ui-radio .ui-btn {
        margin: 0; text-align: left; z-index: 2;
}
```

我们已经介绍了 jQuery Mobile 的主 CSS 文件，现在来看一下 jQuery Mobile 中包含的 5 个调色板，以及它们是如何出现在不同组件上的（见图 7-3 到图 7-6）。

图 7-3 表格调色板

图 7-4 列表调色板

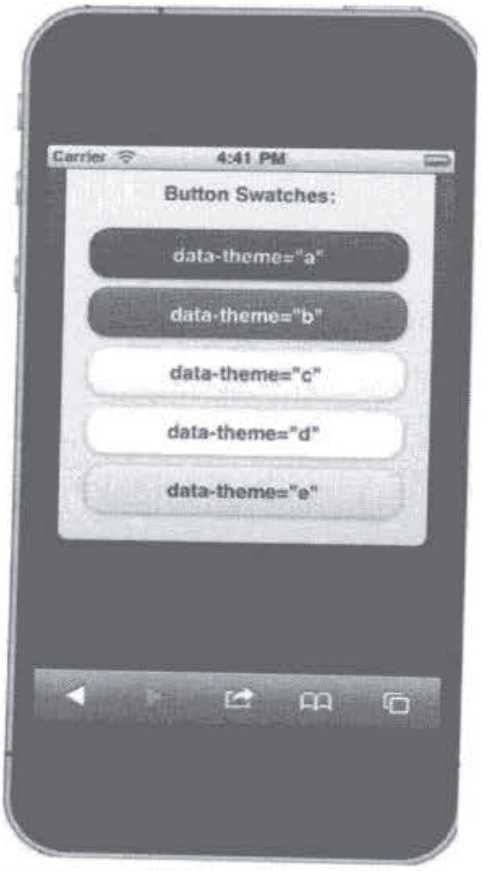


图 7-5 按钮调色板

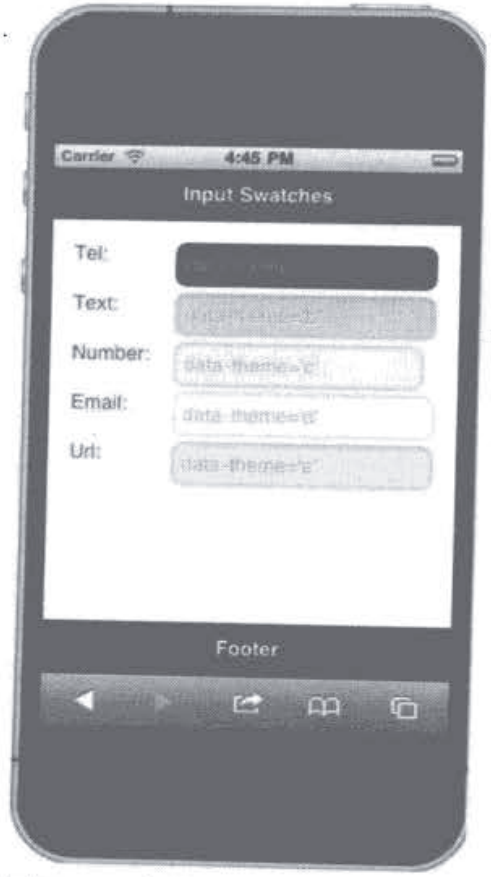


图 7-6 表单字段调色板

为了让调色板的样式在所有的组件上保持一致，需要为每个调色板应用视觉优先级约定，如下所示。

- “a”-（黑色）视觉优先级的最高级别。
- “b”-（蓝色）第二级。
- “c”-（灰色）基线。
- “d”-（白/灰）另外一个（alternate）第二级。
- “e”-（黄色）重色（accent color）。

7.3 主题默认值

如果没有为页面添加 data-theme 属性，jQuery Mobile 会为所有的页面容器和表单元素应用默认的主题（见表 7-1）。

表 7-1 组件的主题

组件	默认主题	是否继承父容器的主题？	示例
按钮	继承自父容器的主题	是	程序清单 4-8
复选框	继承自父容器的主题	是	程序清单 4-20
内容	data-theme=“c”	是	程序清单 7-6
对话框	data-theme=“a”	是	程序清单 2-6
表格	无	是	程序清单 6-3
页脚	data-theme=“a”	否	程序清单 7-6
页眉	data-theme=“a”	否	程序清单 7-6
列表视图	data-theme=“c”	否	程序清单 5-1
列表徽章	data-theme=“c”	否	程序清单 5-9
列表分割线	data-theme=“b”	否	程序清单 5-3
列表条目	data-theme=“c”	是（指从列表继承）	程序清单 7-6
列表拆分按钮	data-theme=“b”	否	程序清单 5-6
页面	data-theme=“c”	否	程序清单 7-5
单选按钮	继承自父容器的主题	是	程序清单 4-18
选择	继承自父容器的主题	是	程序清单 4-16
滑动条	继承自父容器的主题	是	程序清单 4-22
开关	继承自父容器的主题	是	程序清单 4-24
文本输入	继承自父容器的主题	是	程序清单 4-12

例如，如果我们创建了一个基本的 jQuery Mobile 页面，而且没有显式设置其主题，则元素会退回到它们的默认主题，或者是继承它们的父容器的主题。在图 7-7 中，页面、页眉、页脚、内容和列表元素使用的是默认主题，而表单元素使用的是继承的主题。

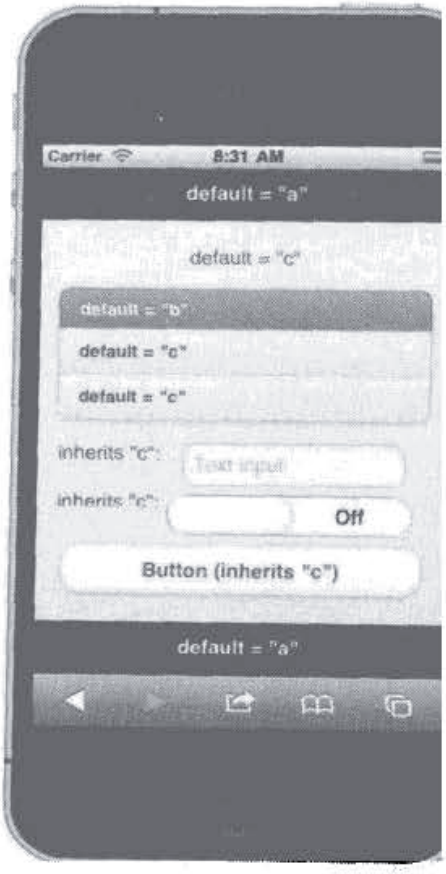


图 7-7 具有默认主题和继承主题的页面

通过参考“组件使用的默认主题”表（见表 7-1），我们可以确定每个组件应用的默认主题是什么。我们进一步查看内容和按钮组件。默认情况下，内容组件会应用 data-theme=“c”。然而，按钮组件没有默认主题，所以它继承了其父容器的默认主题。在程序清单 7-6 中，按钮的父容器是内容组件，因此按钮会继承主题“c”。如果按钮在页眉容器的内部，则会继承页眉容器的主题。

程序清单 7-6 具有默认主题的页面（ch7/theme-defaults.html）

```
<div data-role="page">
    <div data-role="header">
        <h1>default = "a"</h1>
    </div>

    <div data-role="content">
      default = "c"
```

```
        <ul data-role="listview" data-inset="true">
            <li data-role="list-divider">default = "b"</li>
            <li>default = "c"</li>
            <li>default = "c"</li>
        </ul>

        <form id="test" id="test" action="#" method="post">
            <p>
                <label for="text">inherits "c":</label>
                <input type="text" name="text" id="text" value="" />
            </p>
            <p>
                <label for="sound">inherits "c":</label>
                <select name="slider" id="sound" data-role="slider">
                    <option value="off">Off</option>
                    <option value="on">On</option>
                </select>
            </p>

            <a href="#" data-role="button">Button (inherits "c")</a>
        </form>
    </div>

    <div data-role="footer" data-position="fixed">
        <h3>default = "a"</h3>
    </div>
</div>
```

7.4 主题继承

组件也可以继承其父容器的主题。主题继承具有两方面的好处。首先，对设计人员来说，主题继承会让样式化的过程更为高效，这是因为我们可以在一个很高的层级（页面容器）设置一个主题，该主题会级联（cascade）到所有的子组件，从而节省了宝贵的时间。此外，它可以保证组件在整个应用程序中具有一致的样式。例如，在程序清单 7-7 中，我们使用 data-theme="e" 对页面容器进行了样式化。因此，内容主题会从它的父容器那里继承主题“e”（见图 7-8）。

程序清单 7-7 主题继承（ch7/theme-inheritance.html）

```
<div data-role="page" data-theme="e">
        <div data-role="header">
                <h1>No inheritance</h1>
        </div>

        <div data-role="content">
                Inherits "e"

                <ul data-role="listview" data-inset="true">
                        <li data-role="list-divider">No inheritance</li>
                        <li>No inheritance</li>
                        <li>No inheritance</li>
                </ul>
```

```
                <form id="test" id="test" action="#" method="post">
                  <p>
                    <label for="text">Inherits "e"</label>
                    <input type="text" name="text" id="text" value=""/>
                  </p>
                  <p>
                    <label for="sound">Inherits "e"</label>
                    <select name="slider" id="sound" data-role="slider">
                        <option value="off">Off</option>
                        <option value="on">On</option>
                    </select>
                  </p>

                  <a href="#" data-role="button">Button (Inherits "e")</a>
                </form>
        </div>

        <div data-role="footer" data-position="fixed">
                <h3>No inheritance</h3>
        </div>
</div>
```

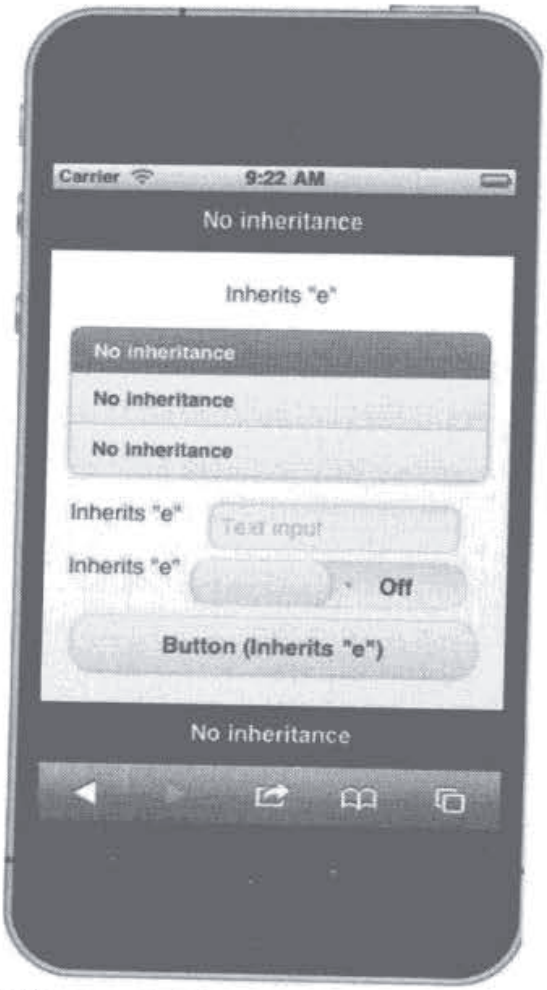


图 7-8 主题继承

> **注意：**并非所有的主题都会继承其容器的主题。有关不会继承其父容器主题的组件列表，请见表 7-1 中的“是否继承其父容器的主题”一栏。

我们也可以为每个组件显式设计主题。设计人员在对站点进行样式化时，采取这种方式会带来灵活性，而且能够构建更为丰富的设计（见图 7-9，相关代码见程序清单 7-8）。

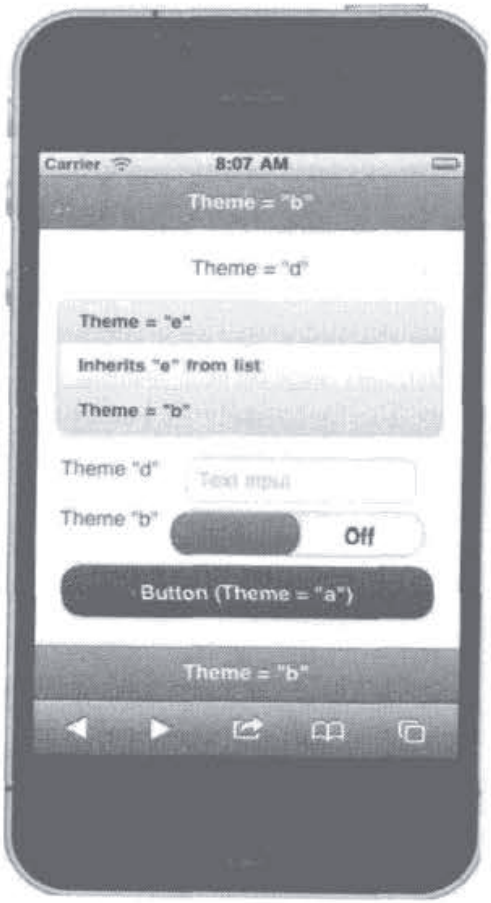


图 7-9 显式的主题

程序清单 7-8 显式的主题（ch7/theme-explicit.html）

```
<div data-role="page" data-theme="e">
    <div data-role="header" data-theme="b">
      <h1>Theme = "b"</h1>
    </div>

    <div data-role="content" data-theme="d">
      Theme = "d"

      <ul data-role="listview" data-theme="e" data-divider-theme="e">
```

```
            <li data-role="list-divider">Theme = "e"</li>
            <li>Inherits "e" from list</li>
            <li data-theme="b">Theme = "b"</li>
        </ul>

        <form id="test" id="test" action="#" method="post">
        <p>
            <label for="text">Theme "d"</label>
            <input type="text" name="text" id="text" data-theme="d" />
        </p>
        <p>
          <label for="sound">Theme "b"</label>
          <select id="sound" data-role="slider" data-theme="b">
             <option value="off">Off</option>
             <option value="on">On</option>
          </select>
        </p>

        <a href="#" data-role="button" data-theme="a">Button</a>
        </form>
      </div>

      <div data-role="footer" data-position="fixed" data-theme="b">
        <h3>Theme = "b"</h3>
      </div>
</div>
```

7.4.1 主题优先级

在为组件应用主题时，需要遵循如下优先级顺序。

1．显式的主题：如果你为任何组件设置了 data-theme 属性，该主题会覆盖任何继承的或默认的主题。

2．继承的主题：继承的主题会覆盖所有默认的主题。例如，在程序清单 7-7 中，内容容器从其页面容器继承来的主题“e”会覆盖掉它的默认主题“c”。有关可以继承其父容器主题的组件列表，请见表 7-1 中的“是否继承父容器的主题”一栏。

3．默认的主题：在没有显式设置主题也没有继承主题时，会应用默认主题。有关组件使用的默认主题列表，请见表 7-1 中的“默认主题”一栏。

注意：默认情况下，内容容器的最小高度只会拉伸（stretch）其内部部件的高度。当内容的主题与其页面容器的主题不同时，这会成为一个问题（见图 7-10），我们可以使用 CSS 来修复这个问题。例如，我们可以将内容容器的最小高度设置为屏幕的高度（见图 7-11）。

```
.ui-content {min-height: inherit; }
```

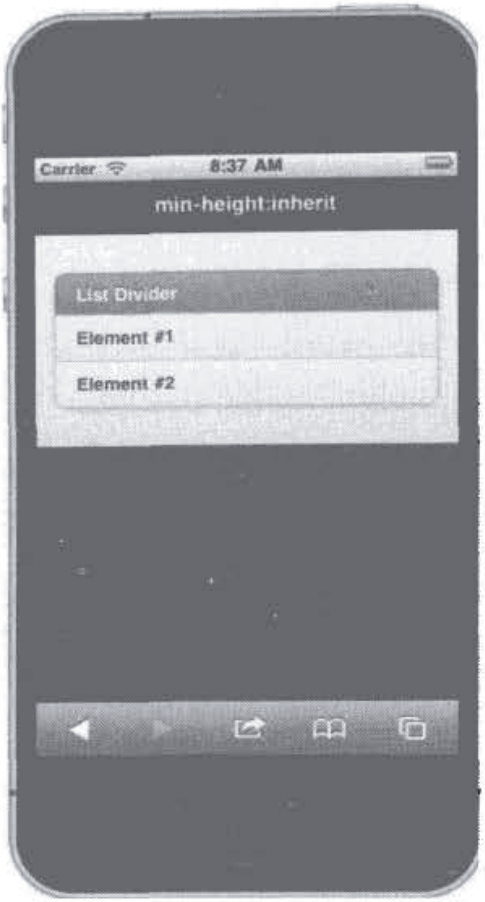


图 7-10 非 100%的内容高度

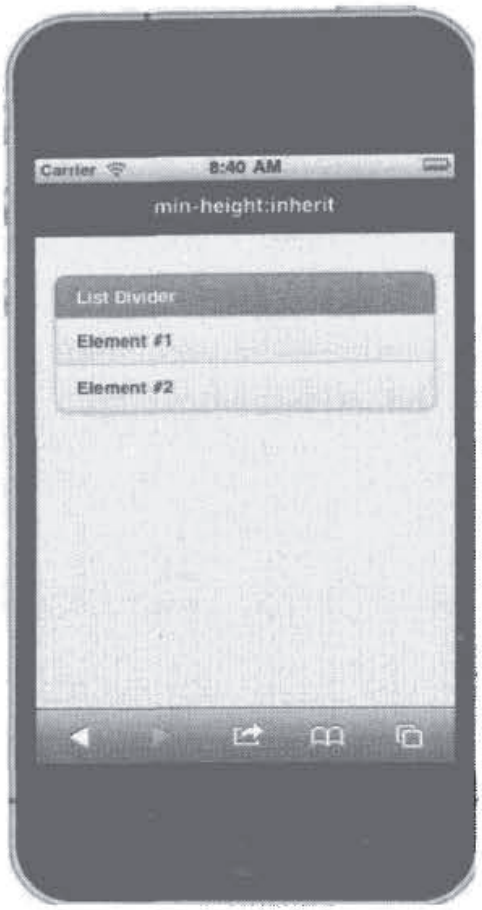


图 7-11 100%的内容高度

7.5 自定义主题

jQuery Mobile 主题框架允许设计人员迅速地自定义或重新样式化他们的用户界面。在本节，我们会学到如何手动创建自定义的调色板。前面讲到，默认的 jQuery Mobile CSS 文档被分为两个部分：主题部分和结构部分。在这个练习中，我们会创建一个自定义的调色板，并将其用于具有潜在风险的动作。例如，常见的用户体验设计指南鼓励开发人员以红色突出显示能够控制潜在危险动作的按钮。在 jQuery Mobile 中，我们可以创建一个自定义调色板，用来管理能够引发危险动作的图标和/或按钮的视觉样式（背景、边界、颜色、字体和阴影）。

要手动创建一个自定义调色板，需要采取如下步骤。

1. 首先，为自定义主题创建一个独立的 CSS 文件（css/theme/custom-theme.cass）。这可以保持自定义文件与主 jQuery Mobile CSS 文件的隔离，而且会简化日后的更新。

注意：如果你计划用自定义主题对整个 jQuery Mobile 应用程序进行样式化，推荐使用从 jQuery Mobile 的下载站点[2]下载的只包含结构的 CSS 文件。这对不需要默认主题的应用程序来说，只是一个轻量级的替换方案，而且能够简化自定义主题的管理（见程序清单 7-9）。

程序清单 7-9 没有默认主题的 jQuery Mobile 结构文件

```
<head>
  <meta charset="utf-8">
  <title>Custom Theme</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel=stylesheet href="css/theme/custom-theme.css" />
  <link rel=stylesheet href="css/structure/jquery.mobile.structure.css"/>
  <script type="text/javascript" src="jquery-min.js"></script>
  <script type="text/javascript" src="jquery.mobile-min.js"></script>
</head>
```

2．寻找一个现有的调色板作为参考的基础。在研究了现有的调色板之后，从中复制与你的新调色板样式最为相似的那个。这可以在最大程度上降低为了创建新调色板而不得不做出的修改次数。对于我的新调色板，我复制了“e”调色板作为我的基础，这是因为“e”调色板是一个重色调色板，而我的新调色板要用于潜在的危险动作，因此可以将我的新调色板归类到重色类别中。

3．接下来，复制基础调色板并粘贴到 custom-theme.css 文件中。然后，重命名该调色板，以便与一个独特的字母（f～z）相关联。例如，将所有带“-e”的 CSS 后缀替换为“-v”（见程序清单 7-10）。现在，需要执行危险动作的任何组件，都可以通过 data-theme=“v”来引用这个新的调色板。

程序清单 7-10 在调色板“e”之后的自定义调色板“v”（ch7/css/theme/custom-theme1.css）

```
/* V
------------------------------------------------------------------------*/
.ui-bar-v {
  font-weight: bold;
  border: 1px solid       #999;
  background:             #dedede;
  color:                  #000;
  text-shadow: 0 1px 0px  #fff;
  -
}
.ui-btn-up-v {
```

[2] 见 http://jquerymobile.com/download.

```
  border: 1px solid       #999;
  background:             #e79696;
  color:                  #fff;
  text-shadow: 0 1px 0px  #fff;
  ...
}
```

4. 现在为我们的新调色板更新 CSS 的视觉设置（背景、边界、颜色、字体和阴影）。对这个新的“v”调色板，我对所有的按钮进行了更新，使其具备一个带白色文本的红色渐变背景（见程序清单 7-11）。

程序清单 7-11 使用红色背景渐变和白色文本来更新“v”调色板（ch7/css/theme/custom-theme1.css）

```
/* V
------------------------------------------------------------------------*/
.ui-btn-up-v {
  border: 1px solid       #999;
  background:             #e79696;
  color:                  #fff;
  text-shadow: 0 1px 0px  #fff;
 background-image: -webkit-gradient(
       linear, 0% 0%, 0% 100%, from(#E79696), to(#ce2021),
       color-stop(.4,#E79696)
  );
  background-image: -webkit-linear-gradient(
       0% 56% 90deg,#CE2021, #E79696, #E79696 100%
  );
  background-image:    -moz-linear-gradient(
       0% 56% 90deg,#CE2021, #E79696, #E79696 100%
  );

 ...
}
```

5. 接下来，将我们的“v”调色板与真实的页面集成，以进行测试。为了测试这个新的“v”调色板，我创建了两个页面。在第一个页面中，我想查看新的调色板如何在只带有图标的按钮上显示。在该测试中，我创建了一个拆分按钮列表，而且其附属按钮作为我们的删除图标，然后使用“v”调色板对附属按钮进行样式化（见图 7-12，相关代码见程序清单 7-12）。

程序清单 7-12 使用“v”调色板的拆分按钮（ch7/custom1.html）

```
<head>
  <link rel=stylesheet href="css/theme/custom-theme1.css" />
  <link rel=stylesheet href="css/structure/jquery.mobile-min.css"/>
  ...
</head>
```

```
<ul data-role="listview" data-split-icon="delete" data-split-theme="v">
    <li>
        <a href="#">
            <img src="../images/kungfupanda2.jpg" />
            <h3>Kung Fu Panda</h3>
            <p>Rated: PG</p>
            <p>Runtime: 95 min.</p>
        </a
     <a href="#delete" data-transition="slidedown">Delete</a>
    </li>
...
```

我还将新的 custom.theme.css 文件导入到主 jQuery Mobile CSS 文件的前面。在第二个测试中，我想在一个删除按钮上应用“v”调色板。在该测试中，我创建了一个对话框来确认潜在危险的动作，并使用新的红色渐变样式主题来样式化删除按钮（见图 7-13，相关代码见程序清单 7-13）。

图 7-12 用于潜在危险动作的红色图标背景

图 7-13 用于删除动作的红色删除按钮

程序清单 7-13 带有“v”调色板的删除按钮（ch7/custom1.html）

```
<div data-role="dialog" id="delete">
    <div data-role="content" data-theme="c">
        <span class="title">Are you sure?</span>

        <a href="#home" data-role="button" data-theme="v">Delete</a>
        <a href="#home" data-role="button" data-rel="back">Cancel</a>
    </div>
</div>
```

最后，在创建新的调色板时，最好是在颜色编码样式指南的要求下对自定义的样式进行归档，这样公司内的所有设计人员和开发人员都会熟悉这些样式的用途。

> **注意：** CSS 渐变生成器[3]是能够自动生成渐变语法的工具，有助于简化前面的步骤 4。

7.6 ThemeRoller

ThemeRoller[4]是一款基于 Web 的工具，能够自动为 jQuery Mobile 生成新的基于 CSS 的主题。这是一种非常有用的工具，因为它允许用户在真实的 jQuery Mobile 布局内，在左边的面板中更新配色方案，并在右面的面板中预览相对应的结果（见图 7-14）。

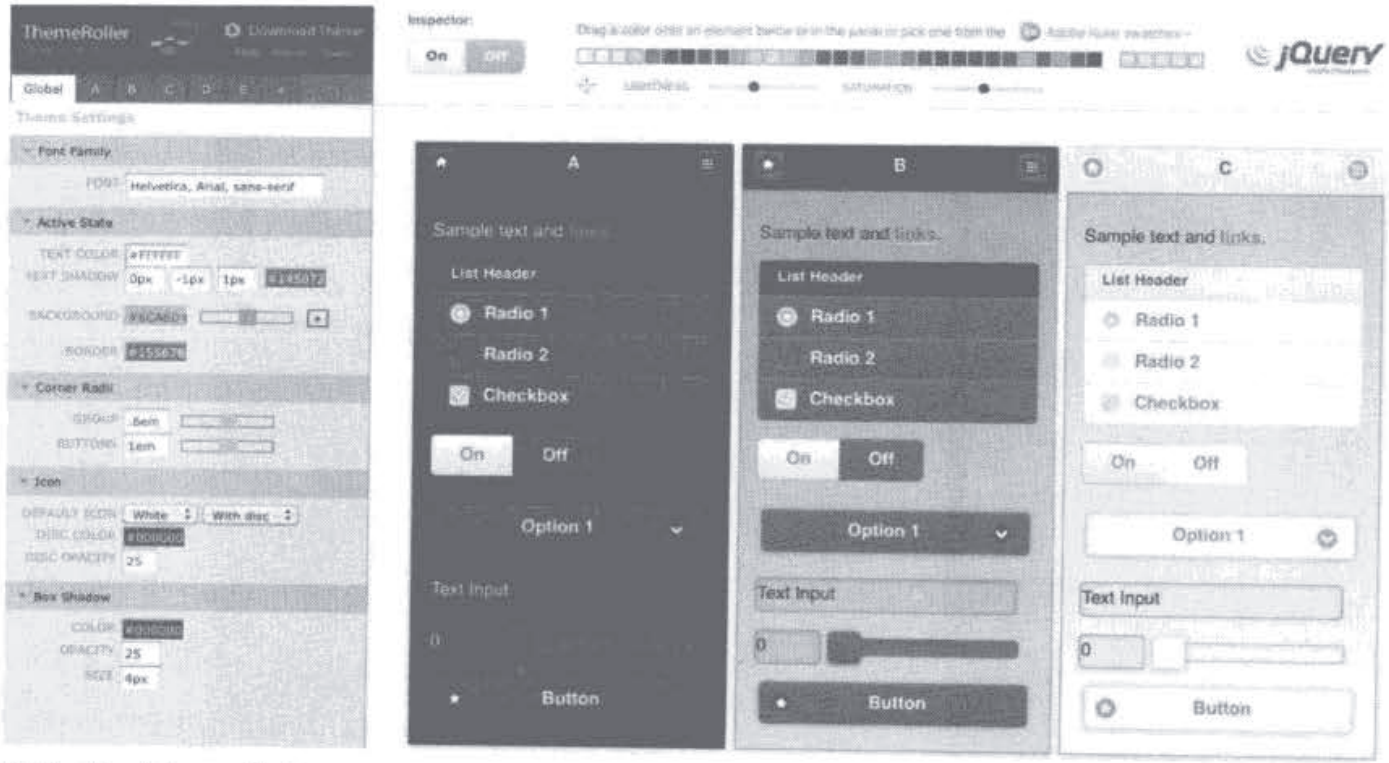


图 7-14 ThemeRoller

7.6.1 调色板和全局设置

在 ThemeRoller 左侧面板中的“Global”选项卡下，你可以迅速调整以全局方式应

3 见 http://www.westciv.com/tools/gradients/或 http://gradients.glrzad.com/.
4 见 http://jquerymobilie.com/themeroller.

用到所有调色板的 CSS 属性。在这里，你可以调整字体集（font family）、活动状态的颜色（active state color）、圆角半径（corner radii）、图标（icon）和阴影（shadow）（见图 7-15）。

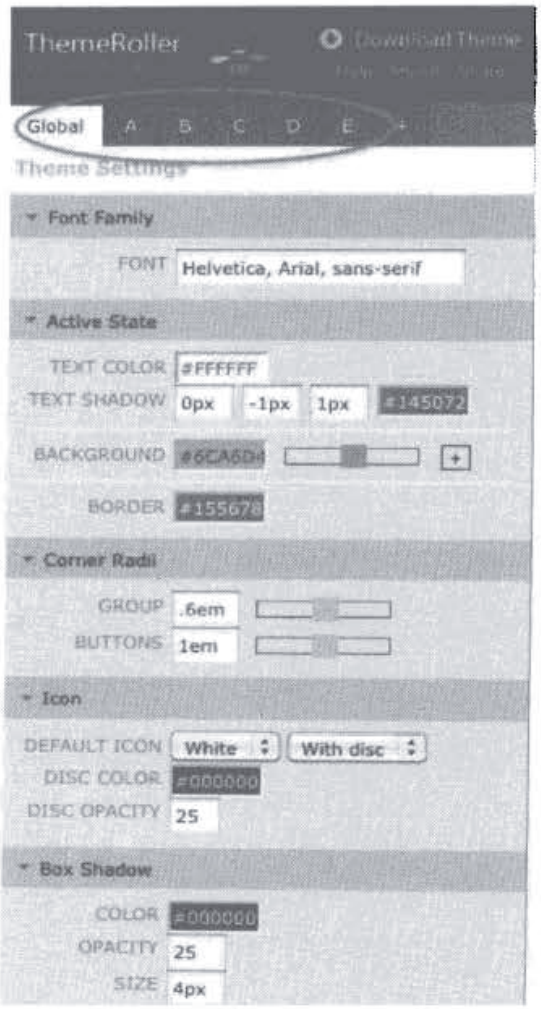


图 7-15 全局主题设置

靠近“Global”选项卡的是具体的调色板选项卡（a～z）。在这里，你可以为你的主题添加、编辑或删除一个调色板（见图 7-15）。

7.6.2 Preview Inspector 和 QuickSwatch Bar

为了更容易地创建自定义主题，在预览面板的顶部有两个独特的工具可供使用：Preview Inspector 和 QuickSwatch Bar。

Preview Inspector 是一个处于“On”或“Off”状态的触发器（toggle）（见图 7-16）。

当触发器为“On”时，在预览面板中单击一个元素会自动在左侧面板中显示该元素的可编辑属性。当需要快速编辑样式时，这样可以节省宝贵的时间。

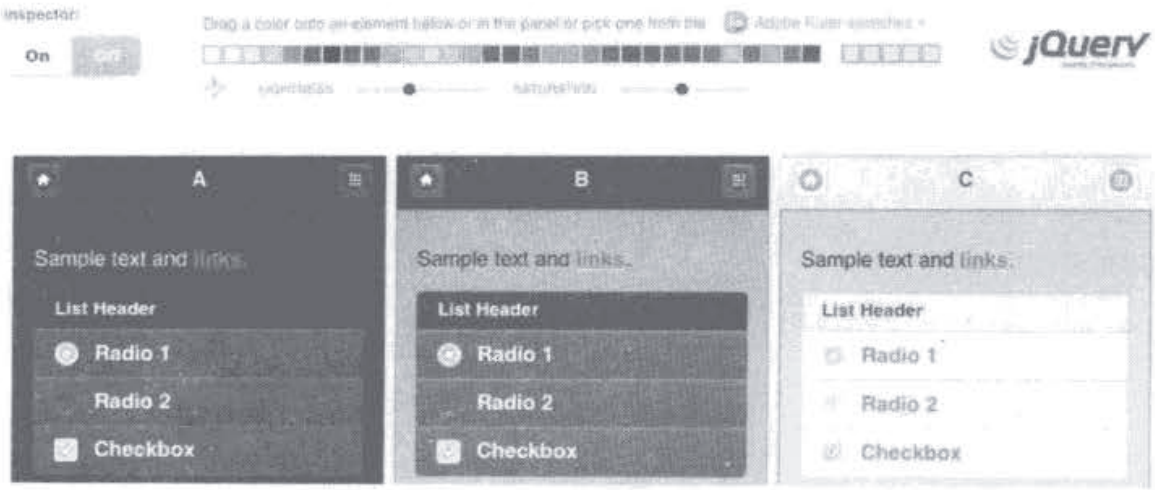


图 7-16 Preview Inspector 和 QuickSwatch Bar

QuickSwatch Bar 是一个出现在 Preview Inspector 右侧的一个色彩频谱（见图 7-16）。这是一个很强大的工具，允许用户将任何颜色拖放到预览页面中的元素上，或者是拖放到左侧面板中的颜色属性上。在 QuickSwatch Bar 的下面是两个滚动条，用来调整调色板（color pallet）的亮度和饱和度。此外，用户最近选择的颜色会显示在色彩频谱的右侧，以方便快速重用。

7.6.3 Adobe Kuler 集成

当用户需要从零开始创建调色板时，可能会遇到麻烦。为了简化该过程，ThemeRolloer 内置了 Adobe 的 Kuler[5]集成工具（见图 7-17）。

图 7-17 Adobe 的 Kuler app

Kuler 是一个允许人们创建、共享调色板，并对调色板进行排名的站点。为了查看

5 见 http://kuler.adobe.com.

Kuler 中可用的调色板，可单击在 QuickSwatch Bar 上方出现的“Adobe Kuler”链接。当 Kuler app 打开时，会在左侧面板中显示一个搜索过滤器，它允许用户按照最近使用的调色板、最流行的调色板，以及调色板排名进行过滤，用户也可以自定义搜索。当你找到感兴趣的一种颜色时，只需将其拖放到预览面板中的元素上即可。

7.6.4 入门

为了便于比较，我会在 ThemeRoller 中创建一个红色的重色调色板（swatch），并将该体验与上一节创建手动调色板时的体验进行比较。在这个练习中，我会使用这个红色的重色调色板来覆盖 jQuery Mobile 的默认“e”调色板。在 ThemeRoller 中，为了更新一个现有的主题，有必要采取如下步骤。

1. 在 ThemeRoller 中，通过单击右上角的“Import”链接导入一个现有的主题（见图 7-18）。对该练习来说，我会导入并修改 jQuery Mobile 的默认主题。

2. 在主题导入之后，找到需要修改的调色板。在该步骤中，我会修改默认的“e”调色板。

3. 接下来，为红色的重色调色板寻找一个合适的基线颜色。我们可以在 QuickSwatch Bar 或 Kuler 集成工具中找到一种合适的红色。

4. 在找到了适当的基线颜色后，现在使用选定的颜色来更新预览面板中的元素。例如，我会使用一个深红的重色对页眉和所有的元素进行样式化。

5. 在预览面板中，进行任何必要的调整。例如，你可能想微调颜色，或者是使用背景渐变添加一些奇妙的效果。可以预料到，与手动的方法相比，ThemeRoller 可以更加高效地处理编辑和预览。

6. 在适应了新主题的布局之后，你可以通过单击 ThemeRoller 右上角的“Download Theme”链接，下载主题的 CSS（见图 7-19）。

图 7-18 导入现有的主题

图 7-19 下载主题

7. 现在可以在应用程序中应用新的主题（见程序清单 7-14，相关截图见图 7-20）。为了简化自定义主题的管理，建议分别载入结构文件和你的自定义主题。

程序清单 7-14 导入 ThemeRoller 中的自定义主题（ch7/custom2.html）

```
<head>
  <meta charset="utf-8">
  <title>Custom Theme</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel=stylesheet href="css/theme/custom-theme2.css" />
  <link rel=stylesheet href="css/structure/jquery.mobile.structure.css"/>
  <script type="text/javascript" src="jquery-min.js"></script>
  <script type="text/javascript" src="jquery.mobile-min.js"></script>
</head>

<div data-role="dialog" id="delete">
    <div data-role="content" data-theme="c">
        <span class="title">Are you sure?</span>

        <a href="#home" data-role="button" data-theme="e">Delete</a>
        <a href="#home" data-role="button" data-rel="back">Cancel</a>
    </div>
</div>
```

图 7-20 ThemeRoller 的红色删除按钮

7.7 总结

jQuery Mobile 主题框架是一个面向对象的 CSS3 框架，它具有轻量级、可自定义的特点，能够在所有的浏览器上呈现出统一的设计。在本章中，我们讲解了主题框架的基础知识，以及 jQuery Mobile 中包含的 5 个调色板。

我们还讲解了将主题分配为组件的三种方式。尽管可以使用 data-theme 属性显式地设置所有组件的主题，但是大多数组件都有默认的主题，而且也可以继承其父容器的主题。我们查看了这三种方式的示例，并讲解了将主题分配给组件时应该遵循的优先顺序。

最后，我们讲解了如何创建自定义的调色板。无论是需要创建一个更为丰富的设计，还是需要创建一个最能匹配公司品牌文化的设计，灵活的主体化框架都能满足所有的要求。我们还讲解了用来创建自定义调色板的两种方法，以及这两种方法对应的示例。我们还讲到，手动创建调色板的方式能够让我们完全控制布局，而 jQuery Mobile 的新 ThemeRoller 则提供了一种更直观而且高效的工作环境。

在下一章，我们会深入讲解 jQuery Mobile API。我们会讲到如何配置 jQuery Mobile，还会讲到 API 的核心方法、事件、属性（property）和数据属性（attribute）。

第8章

jQuery Mobile API

所有精心编写的框架都允许开发人员扩展和覆盖默认的配置设置。此外，它们还提供了简便的方法来简化代码。jQuery Mobile 包含一个相当强大的 API，这个 API 包含了所有这些简便的特性。首先，我们会讲解如何配置 jQuery Mobile。我们会讲解 jQuery Mobile 内的每一个特性，重点讲解它的默认设置，并演示如何使用 API 来配置每一个选项。然后，我们会讲解 jQuery Mobile 所具有的最受欢迎的方法、页面事件和属性。在通过程序方式更新你的移动 Web 应用程序时，这些 API 特性会相当有用。最后，我们会讲解一个列出了所有 jQuery Mobile 数据属性的已排序表格。对每个属性，都会给出简单描述、示例和它增强的组件示意图。

8.1 配置 jQuery Mobile

在 jQuery Mobile 初始化时，它会在 document 对象上触发一个 mobileinit 事件。可以绑定到 mobileinit 事件，然后应用对 jQuery Mobile 的（$.mobile）默认配置设置的覆盖。此外，可以使用额外的行为和属性来扩展 jQuery Mobile。例如，有两种方式可以用来配置 jQuery Mobile，如下面的示例所示。你可以通过 jQuery 的 extend 方法来覆盖属性，也可以单独进行覆盖。

示例：

```
// Configure properties via jQuery's extend method
$( document ).bind( "mobileinit", function(){
  $.extend( $.mobile, {
    // Override loading message
    loadingMessage: "Loading...",
```

```
    // Override default transition from "slide" to "pop"
    defaultTransition: "pop"
  });
});
// Configure properties individually
$( document ).bind( "mobileinit", function(){
  $.mobile.loadingMessage = "Initializing";
  $.mobile.defaultTransition = "slideup";
});
```

8.1.1 自定义脚本的位置

由于在执行 jQuery Mobile 时，会立即触发 mobileinit，因此需要将自定义脚本放在 jQuery Mobile JavaScript 文件之前。

示例：

```
<head>
    <script type="text/javascript" src="jquery-min.js"></script>
    <script type="text/javascript" src="custom-scripts-here.js"></script>
    <script type="text/javascript" src="jquery.mobile-min.js"></script>
</head>
```

8.1.2 可配置的 jQuery Mobile 选项

下面是可配置的$.moible 选项，你可以在你的自定义 JavaScript 内对其进行覆盖。

- activeBtnClass(string, default: "ui-btnactive")

用来识别和样式化“活动”按钮的 CSS 类。这个 CSS 属性通常用来样式化和识别标签栏中的活动按钮。

- activePageClass(string, default: "ui-page-active")

这个 CSS 类分配给当前可见和活动的页面或对话框。例如，当多个页面载入到 DOM 中时，活动的页面会应用这个 CSS 属性。

- ajaxEnabled(boolean, default: true)

在可能的情况下，通过 Ajax 动态载入页面。默认情况下，所有页面的 Ajax 载入都是打开的，但是外部 URL、使用 rel="external" 或 target="_blank" 属性标记的链接除外。如果禁用 Ajax，页面链接会使用普通的 HTTP 请求载入，而且不会用到 CSS

转换。

■ allowCrossDomainPages(boolean, default: false)

在使用 PhoneGap 进行开发时，建议将该配置选项设置为 true。这会允许 jQuery Mobile 管理 PhoneGap 中跨域（cross-domain）请求的页面载入逻辑。

■ autoInitializePage(boolean, default: true)

对于想要完全控制页面初始化顺序的高级开发人员来说，可以将该配置选项设置为 false，这会禁用所有页面组件的自动初始化。这使得开发人员能够根据需要手动增强每一个组件。

■ defaultDialogTransition(string, default: "pop")

在转换到一个对话框时，使用的默认转换。如果不需要转换，可以将该转换设置为“none”。

■ defaultPageTransition(string, default: "slide")

在转换到一个页面时，使用的默认转换。如果不需要转换，可以将该转换设置为“none”。

■ gradeA（返回一个布尔值的函数，default: 浏览器必须支持媒体查询或者支持 IE 7 以及更高版本）

jQuery Mobile 会调用该方法来确定框架是否应用了动态的 CSS 页面增强。默认情况下，该方法会为支持媒体查询的所有浏览器应用增强。但是，jQuery Mobile 只会增强 A 级浏览器的页面。IE 7 以及更高版本属于 A 级浏览器，因此它们的显示也会被增强。例如，$.mobile.gradeA 的当前函数如下所示。

$.mobile.gradeA:

```
$.mobile.gradeA: function(){
  return $.support.mediaquery ||
         $.mobile.browser.ie && $.mobile.browser.ie >= 7;
}
```

■ hashListeningEnabled(boolean, default: true)

基于 location.hash 自动载入和显示页面。jQuery Mobile 监听 location.hash 的改变，以载入 DOM 内的内部页面。你可以禁用该选项，通过手动方式来处理 hash 的改变：

也可以禁用该选项，以访问作为深链接的锚的书签。

■ loadingMessage(string, default: “loading”)

设置载入消息，使其在基于 Ajax 的请求期间出现。此外，可以指派一个 false（boolean）来禁用该消息。如果你想在运行时基于每个页面来更新载入消息，则可以在页面内对其进行更新。

示例：

```
// Update loading message
$.mobile.loadingMessage = "My custom message!";

// Show loading message
$.mobile.showPageLoadingMsg();
```

■ minScrollBack(string, default: 250)

设置最小的滚动距离，而且在返回页面时，该值也能被记住。在返回一个页面时，如果链接的滚动位置超出了 minSrollBack 的设置，则框架会自动滚动到启动转换的位置或链接。默认情况下，滚动阈值是 250 像素，如果你希望删除这个最小的设置，以便框架在滚动时能够无视滚动的位置，则可以将该值设置为 0。如果想要禁用该特性，则将其值设置为“infinity”。

■ nonHistorySelectors(string, default: “dialog”)

你可以指定将哪个页面组件排除在浏览器的历史记录栈之外。默认情况下，带有 data-rel=“dialog”的任何链接，或者是带有 data-role=“dialog”的任何页面都不会出现在历史记录中。此外，在导航到相应的页面时，这些非历史的选择器组件也不会更新它们的 URL，这样做的结果是无法为这些页面添加书签。

■ ns(string, default:“”)

用于 jQuery Mobile 内自定义 data-*属性的名称空间。在 HTML5 内，数据属性属于新特性。例如，“data-role”是 role 属性的默认名称空间。如果你想要以全局方式覆盖默认的名称空间，则需要覆盖$.mobile.ns 选项。

示例：

```
// Set a custom namespace
$.mobile.ns = "jqm-";
```

这样做的结果是，所有的 jQuery Mobile data-*属性都需要前缀“data-jqm-”。例如，“data-role”属性现在变成“data-jqm-role”。

重要：如果要更新默认的名称空间，则需要更新在 jQuery Mobile CSS 文件内发现的一个 CSS 选择器：

```
// Original CSS for default namespace:
.ui-mobile [data-role=page],
.ui-mobile [data-role=dialog],
.ui-page {..}

// Updated CSS for the new namespace "jqm-":
.ui-mobile [data-jqm-role=page],
.ui-mobile [data-jqm-role=dialog],
.ui-page {..}
```

为什么要覆盖默认的名称空间

首先，如果你在设计一个包含 HTML5 data-*属性的 JavaScript 框架，W3C 建议你在其中包含一个钩子（hook），以允许开发人员自定义名称空间，从而避免与第三方框架发生冲突。当遇到与第三方框架发生的冲突时，需要改变你的默认名称空间。

■ page.prototype.options.addBackBtn(Boolean, default: false)

如果希望某个应用程序上显示回退按钮，则将该选项设置为 true。jQuery Mobile 内的回退按钮是一个智能的微件。只有当要回退的页面处于历史记录栈中时，回退按钮才会显示。

示例：

```
$.mobile.page.prototype.options.addBackBtn = true;
```

■ page.prototype.options.keepNative(string. default:
`:jqmData(role='none');:jqmData(role='nojs')"`

如果希望在无需为标记添加 data-role=“none”的情况下阻止自动初始化，可以自定义用来阻止自动初始化的 keepNative 选择器。例如，为了阻止框架初始化所有的选择和输入元素，我们可以更新该选择器。

示例：

```
$.mobile.page.prototype.options.keepNative = "select, input";
```

■ pageLoadErrorMessage(string, default: “Error Loading Page”)

当一个 Ajax 页面请求载入失败时，会出现该错误响应消息。

■ subPageUrlKey(string, default: “ui-page”)

用来引用由微件生成的子页面的 URL 参数。子页面 URL 的一个例子是“nested-list.html&ui-page=Movies-3”。嵌套的列表视图是一个特殊的微件，它将每个列表分割为独立的子页面。例如，刚才显示的 URL 具有一个“Movies”子列表，而且 jQuery Mobile 会将该子列表转换为它自己的子页面，以供深链接引用。如果需要重命名这个 URL 参数，可以使用$.mobile.subPageUrlKey 来更改。

■ touchOverflowEnabled(boolean, default: false)

为了使用本地的惯性滚动（momentum scrolling）来实现真正固定的工具栏，浏览器需要支持两种定位：fixed 或 overflow：auto。幸运的是，新发布的 WebKit（iOS5）开始支持该行为。该选项很有可能在将来成为默认启用的。不过在该事实发生之前，我们可以通过将该配置选项设置为 true，来启用该行为。

8.2 方法

jQuery Mobile 提供了一套方法，当需要以程序方式来更新你的移动 Web 应用程序时，将会用到这套方法。

■ $.mobile.changePage()

changePage 函数处理页面相互转换时涉及的所有细节。

用途：

```
$.mobile.changePage(toPage, [options])
```

参数：

■ toPage（string 或 jQuery 集）。要转化到的目标页面。

- toPage（string）。文件 URL（“contact.html”）或内部元素的 ID（“#contact”）。
- toPage(object)。包含一个页面元素，并将该页面元素作为第一个参数的 jQuery 集对象：$（“#contactPage”）。

options（object）。配置 changPage 请求的一组键/值对。所有的设置都是可选的。

- allowSamePageTransiton（boolean，default: false）。changePage 方法会忽略转到同一页面的请求。将该选项设置为 true，则允许转换到同一个页面。
- chageHash（Boolean，default: true）。当页面转换完成时，更新 toPage 的 URL 的 hash。
- data（string 或 object，default：undefined）。发送到 Ajax 页面请求的数据。
- dataUrl（string，default: toPage URL）。对 URL 进行设置，使其在浏览器的地址栏显示。
- fromHashChange（Boolean，default:false）。指示 chagnePage 是否来自于一个 hashchange 事件。
- fromPage（string，default: $.mobile.activePage）。指定 from 页面。
- pageContainer（jQuery 集，default:$.mobile.pageContainer）。在页面载入之后，指定应该包含该页面的元素。
- reloadPage（Boolean，default:false）。强制重新载入一个页面，即使该页面已经位于页面容器的 DOM 中。
- reverse（boolean，default：false）。指示是向前转换还是后退转换。默认的转换是向前转换。
- role（string，default:“page”）。在显示页面时使用的 data-role 的值。对话框使用的是“dialog”。
- showLoadMsg（Boolean，default:true）。在请求一个页面时，显示载入消息。
- transition（string，default:$.mobile.defaultTransition）。应用到 changepage 的转换。默认的转换是滑动转换。
- type(string, default:“get”)。指定在生成一个页面请求时所使用的方法（“get”

或“post”)。

示例#1:

```
//Transition to the "contact.html" page.
$.mobile.changePage( "contact.html" );

<!-- Markup equivalent when link clicked -->
<a href="contact.html">Contact Us</a>
```

示例#2:

```
// Transition to the internal "#contact" page with a reverse "pop" transition.
$.mobile.changePage( "#contact", { transition: "pop", reverse: true } );

<!-- Markup equivalent when link clicked -->
<a href="contact.html" data-transition="pop" data-
direction="reverse">Contact</a>
```

示例#3:

```
/* Dynamically create a new page and open it */

// Create page markup
var newPage = $("<div data-role=page data-url=hi><div data-role=header>
    <h1>Hi</h1></div><div data-role=content>Hello Again!</div></div>");

// Add page to page container
newPage.appendTo( $.mobile.pageContainer );

// Enhance and open new page
$.mobile.changePage( newPage );
```

■ $.mobile.hidePageLoadingMsg()

移除或隐藏页面载入消息($.mobile.loadingMessage)。默认的载入消息是“载入”,而且这是可以配置的。要显示载入消息,请见$.mobile.showPageLoadingMsg()。

示例:

```
// Remove the loading message
$.mobile.hidePageLoadingMsg();
```

■ $.mobile.loadPage()

loadPage 函数将一个页面载入到当前页面的 DOM 中,并对其增强。该方法也可以用作一个数据属性,而且可以附加到链接或按钮中(见“数据属性”一节的“data-prefetch”)。

用途:

```
$.mobile.loadPage(url, [options])
```

参数：

url（string）。要载入的页面。

- url（string）。文件 URL（“contact.html”）

options（object）。配置 changePage 请求的一组键/值对。所有的设置都是可选的。

- data（string 或 object，default：undefiend）。要发送到 Ajax 页面请求的数据。
- loadMsgDelay（number（in ms），default：50）。在显示载入消息之前，添加一个动的延迟。该延迟允许框架在无需一条载入消息的情况下就能够载入一个缓存的页面。
- PageContainer（jQuery 集，default：$.mobile.pageContainer）。在页面载入之后，应该包含该页面的元素。
- reloadPage（Boolean，default:false）。强制重新载入一个页面，即使该也卖弄已经位于页面容器的 DOM 中。
- role（string，default:@data-role attribute）。载入页面是需要的 data-role。默认值是元素定义的@data-role 属性。
- showLoadMsg（boolean，default：true）。在请求一个页面时，显示载入消息。
- type（string，default：“get”）。指定在生成一个页面请求时所使用的方法（“get”或“post”）。

示例：

```
// Dynamically load a page and transition to it.
$.mobile.loadPage("page1.html" );

$.mobile.changePage("#page1" ); // data-url value
```

- $.mobile.showPageLoadingMsg()

显示页面载入消息（$.mobile.loadingMessage）。

示例：

```
// Show the page loading message
$.mobile.showPageLoadingMsg();
```

- $.mobile.silentScroll(number)

垂直滚动页面。在框架内，只要一个页面被恢复，就会调用 silentScroll。例如，当单击回退按钮时，silentScroll 方法会在前一个页面显示之前被触发，而且会恢复前一个页面的滚动位置。焦点会出现在触发该初始转换的组件上。在 silentScroll 期间，不会触发 scrollstart 和 scrollstop 事件。

示例：

```
// Hide the iOS address bar
$.mobile.silentScroll(0);

// Scroll down 400 pixels
$.mobile.silentScroll(400);
```

■ $.jqmData()

这是 jQuery.data()方法[1]的移动版本。该方法提供了$.data()内的所有功能，而且能够确保使用 jQuery Mobile 的数据名称空间（$.mobile.ns）来设置和获取所有的数据。

示例：

```
// Find all pages (data-role="page") in the DOM via a selector.
var $pages = $( ":jqmData(role='page')" );

// Find the theme (data-theme) for the first page
var firstPage = $pages.first();
var theme = $.jqmData( firstPage, "theme" );
```

■ $.jqmHasData()

这是 jQuery.hasData()方法[2]的移动版本。该方法提供了$.hasData()内的所有功能，而且能够确保使用 jQuery Mobile 的数据名称空间（$.mobile.ns）来获取所有的数据。

示例：

```
// Does a theme exist for the first page
var hasTheme = $.jqmHasData( firstPage, "theme" );
```

■ $.jqmRemoveData()

这是 jQuery.removeData()方法[3]的移动版本。该方法提供了$.removeData()内的所有功能，而且能够确保使用 jQuery Mobile 的数据名称空间（$.mobile.ns）来删除所有的数据。

---

1 见 http://appi.jquery.com/jQuery.data.
2 见 http://api.jquery.com/jQuery.hasData/.
3 见 http://api.jquery.com/jQuery.removeData/.

示例：

```
// Set data on the first page
$.jqmData(firstPage, "testData", "testValue");

// Remove the data from the first page
$.jqmRemoveData( firstPage, "testData" );
```

8.3 事件

jQuery Mobile 还提供了多个有用的事件，你可以通过编程方式来使用这些事件，以便在移动 Web 应用程序内的页面变化期间，应用预处理过程或事后处理过程。在本节，我们会讲解可在自己的代码中使用的所有 jQuery Mobile 页面事件。在介绍 jQuery Mobile 事件之前，我们先来看一个示意图（见图 8-1）。该示意图显示了 jQuery Mobile 内发生的主要页面事件，该图有助于描述在页面变化生命周期内，每个事件的先后顺序。

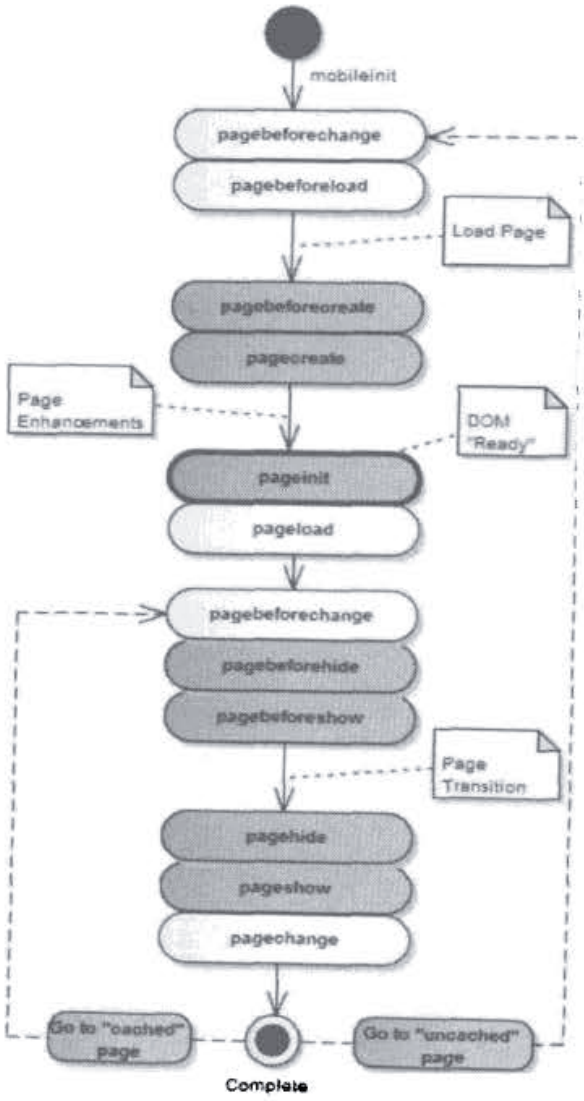


图 8-1 jQuery Mobile 的页面事件

我们已经看到了在页面变化生命周期内，页面事件的触发顺序，先来看每一个具体页面事件的详情。

8.3.1 事件概览

- mobileinit

在 jQuery Mobile 初始化时，它在 document 对象上触发一个 mobileinit 事件。你可以绑定到 mobileinit 事件，然后应用对 jQuery Mobile 的默认配置的覆盖。绑定到 mobileinit 事件的示例，请见“配置 jQuery Mobile”一节。

1. 页面改变事件

在导航到另外一个页面时，会自动在文档上触发页面改变事件。从内部运行机制来看，当调用$.mobile.changePage 方法时，会触发这些事件。在该进程期间，会发生两个事件。第一个触发的事件是 pageforechange。第二个事件则依赖于页面改变的状态。当页面改变成功时，pagechange 事件会被触发，如果页面改变事变，则 pagechangefailed 时间被触发。

- Pagebeforechange

这是在页面改变期间触发的第一个事件。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是一个数据对象。通过调用事件的 preventDefault，可以取消页面改变。此外，通过检查和更新数据对象，可以覆盖页面改变。作为第二个参数传递的数据对象包含如下属性。

- toPage（string）。一个文件 URL 或一个 jQuery 集对象。这与传递给$.mobile.changePage()的参数相同。
- options（object）。这与传递给$.mobile.changePage 的选项相同。

示例：

```
$( document ).bind( "pagebeforechange", function( e, data ) {
    console.log("Change page starting...");

    // Get the page
    var toPage = data.toPage;

    // Get the page options
    var options = data.options;
```

```
    // Inspect toPage or override options (redirect)…

    // Prevent a page change
    e.preventDefault();
});
```

■ pagechange

这是在页面成功改变之后触发的最后一个事件。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是数据对象。作为第二个参数传递的数据对象包含如下属性。

- toPage(string)。一个文件 URL 或一个 jQuery 集对象。这与传递给$.mobile.changePage()的参数相同。
- options(object)。这与传递给$.mobile.changePage 的选项相同。

示例：

```
$( document ).bind( "pagechange", function( e, data ){
    console.log("Change page successfully completed...");
    var toPage = data.toPage;
    var options = data.options;
});
```

■ pagechangefailed

在页面更改失败时，会触发该事件。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是数据对象。作为第二个参数传递的数据对象包含如下属性。

- toPage(string)。一个文件 URL 或一个 jQuery 集对象。这与传递给$.mobile.changePage()的参数相同。
- options(object)。这与传递给$.mobile.changePage 的选项相同。

示例：

```
$( document ).bind( "pagechangefailed", function( e, data ){
    console.log("Page change failed...");
});
```

2. 页面载入事件

当框架将一个页面载入到 DOM 中时，会在文档上触发页面载入事件。从程序角

度来讲，当调用$.mobile.loadPage 时，会触发该事件。在该进程期间，loadPage()会触发两个事件。第一个事件是 pagebeforeload，第二个事件是 pageload 事件（页面载入成功时）或 pageloadfailed 事件（页面载入失败时）。

- pagebeforeload

这是页面载入期间触发的第一个事件。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是数据对象。你可以手动处理载入逻辑。为了实现该操作，必须调用事件的 preventDefault()，并调用延迟对象引用（deferred object reference）的 resolve()或 reject()方法，其中延迟对象应用包含在数据对象中。作为第二个参数传递的数据对象包含如下属性。

- url(string)。发送给$.mobile.loadPage()的相对 URL。
- absUrl(string)。URL 的完全引用。
- deataUrl(string)。实际存储在页面 data-url 属性中的 URL 版本。该 URL 显示在浏览器的地址栏中。
- deferred(object)。调用 preventDefault()来手动处理页面载入时，必须调用该对象的 resolve()或 reject()，以便 changePage()请求能够恢复处理进程。
- options(object)。这与传递给$.mobile.loadPage 的选项参数相同。

示例：

```
$( document ).bind( "pagebeforeload", function( e, data ){
    console.log("Page load starting…");

    // Let the framework know we're manually loading the page
    e.preventDefault();

    // Manually load the document and insert it into the DOM
    var response = manuallyLoadPage();

    if (response.status = "success") {
      // Call resolve passing in the url, options, and jQuery
      // collection object containing the DOM element for the page
      data.deferred.resolve( data.absUrl, data.options,
          response.page);
    } else {
      // The load failed, call reject
      data.deferred.reject( data.absUrl, data.options );
    }
});
```

■ pageload

当页面成功载入到 DOM 中时，会触发该事件。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是数据对象。作为第二个参数传递的数据对象包含如下属性。

- url（string）。发送给$.mobile.loadPage()的相对 URL。
- absUrl（string）。URL 的绝对引用。
- dataUrl（string）。实际存储在页面 data-url 属性中的 URL 版本。该 URL 显示在浏览器的地址栏中。
- options（object）。这与传递给$.mobile.loadPage()的选项参数相同。

示例：

```
$( document ).bind( "pageload", function( e, data ){
  console.log("Page successfully loaded into DOM...");
});
```

■ pageloadfailed

当页面载入失败时，会触发该事件。在该过程期间，框架会显示一条页面载入失败消息，并调用延迟对象的 reject()。通过调用事件的 preventDefault()，在回调时能够防止该默认行为被执行。

示例：

```
$( document ).bind( "pageloadfailed", function( e, data ){
  console.log("Page load failed...");
});
```

3. 页面初始化事件

在 jQuery Mobile 增强页面之前和之后，会触发页面初始化事件。你可以绑定到这些事件，以便在框架增强页面之前对标记进行预解析，或者是在框架增强页面之后设置 DOM ready 事件处理程序。在页面的生命周期之内，这些事件只被触发一次。

■ pagebeforecreate

在页面改变期间，该事件在正在进行初始化的页面上触发。当页面容器已经被插入到 DOM 中之后，但是在页面被增强之前，该事件才发生。在框架增强页面之前

这是预解析标记的首选位置。例如，在该事件中，你能够动态创建和添加虚拟的页面微件，或者是修改现有的数据属性。

示例：

```
$( "#to-page-id" ).live( "pagebeforecreate", function(){
console.log( "Pre-parse the markup before the framework enhances the widgets" );
});
```

■ pagecreate

在页面改变期间，该事件在正在进行初始化的页面上触发。这是由框架触发的事件，用来初始化所有的页面插件。如果你创建了自定义的页面插件，这将是对这些插件进行初始化的首选位置。

示例：

```
$( "#to-page-id" ).live( "pagecreate", function(){
    console.log("Page plugins are being initialized...");

    // Initialize custom plugins
    ( ":jqmData(role='my-plugin')" ).myPlugin();
});
```

■ pageinit

在页面增强结束之后，该事件在正在初始化的页面上发生。该页面现在处于 DOM ready 状态。

示例：

```
$( "#to-page-id" ).live( "pageinit", function(){
    console.log("The page has been enhanced...");
    // Attach event handlers or run other jQuery code...
});
```

4. 页面转换事件

在页面转换期间，页面转换事件在“from”和“to”页面上被触发。当页面在视图中显示或从视图中移除时，可以与这些事件绑定，以进行观察。

■ pagebeforehide

在转换开始时，在“from”页面上触发。该事件在 pagebeforeshow 事件之前发生，而且只有当页面更改请求具有相关联的“from”页面时才能触发。回调该事件时，会

传递两个参数。第一个参数是事件，第二个参数是数据对象。作为第二个参数传递的数据对象包含如下属性。

- nextPage(object)。一个包含要转换到的页面元素的 jQuery 集对象。

示例：

```
$( "#from-page-id" ).live( "pagebeforehide", function( e, data ){
  console.log( "The page transition is just starting..." );
});
```

- pagebeforeshow

在页面被增强之后，并且在页面转换开始之前，该事件在“to”页面上触发。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是数据对象。作为第二个参数传递的数据对象包含如下属性。

- prePage（object）。一个包含转换之前的页面元素的 jQuery 集对象。

示例：

```
$( "#to-page-id" ).live( "pagebeforeshow", function( e, data ){
  console.log( "The page transition is just starting..." );
});
```

- pagehide

在页面转换完成之后，并且在 pageshow 事件之前，该事件在“from”页面上触发，而且只有当页面更改请求具有相关联的“from”页面时才能触发。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是数据对象。作为第二个参数传递的数据对象包含如下属性。

- nextPage（object）。一个包含要转换到的页面元素的 jQuery 集对象。

示例：

```
$( "#from-page-id" ).live( "pagehide", function( e, data ){
  console.log( "The page transition is complete!" );
});
```

- pageshow

在页面转换完成之后，并且在“from”页面被隐藏之后，该事件在“to”页面上触发。回调该事件时，会传递两个参数。第一个参数是事件，第二个参数是数据对象

作为第二个参数传递的数据对象包含如下属性。

- prevPage（object）。一个包含转换之前的页面元素的 jQuery 集对象。

示例：

```
$( "#to-page-id" ).live( "pageshow", function( e, data ){
  console.log( "The page transition is complete!" );
});
```

注意： jQuery Mobile 团队已经创建了一些有用的书签，以允许用户从浏览器控制台查看页面事件历史（见图 8-2）。在导航 jQuery Mobile 应用程序时，可以按照页面、URL 和时间戳这 3 种方式来查看事件历史。要安装这些书签，请到 jQuery Mobile 事件记录页面[1]，然后按照指示进行安装。

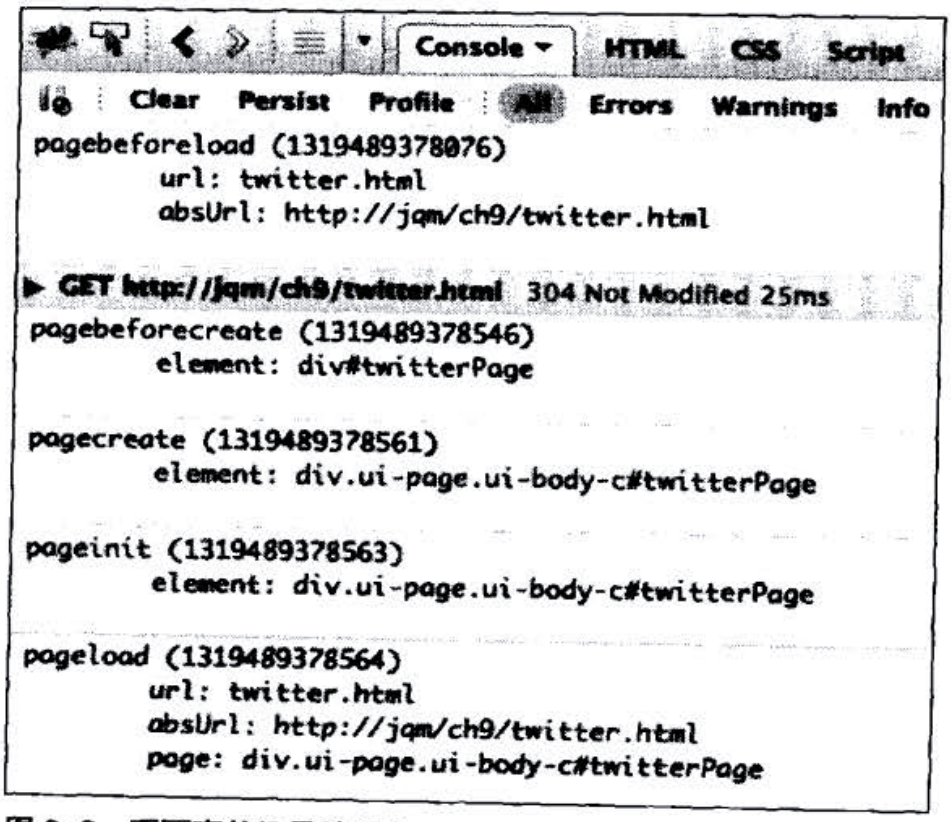


图 8-2 页面事件记录控制台

8.3.2 触发事件

在构建动态页面时，触发 jQuery Mobile 页面事件会比较有用。例如，如果需要为

[1] 见 http://jquerymobile.com/test/tools/log-page-events.html.

页面添加多个新的组件，你可以调用 create 事件，以同时增强所有的新微件。

■ trigger("create")

我们可以触发这个事件，以自动增强页面上的所有新元素。该事件在页面控制器上被触发。

示例：

```
// Add two new buttons to the page
$( '<button id="b2">Button2</button>' ).insertAfter( "#b1" );
$( '<button id="b3">Button3</button>' ).insertAfter( "#b2" );

// Enhance the new buttons on the page
$.mobile.pageContainer.trigger( "create" );
```

8.4 属性

jQuery Mobile 也有一组可供公众使用的属性，这样，你无需编写自己的 jQuery Mobile 选择器就可以访问常见的组件了。

■ $.mobile.activePage

获得当前处于活动状态或者可见状态的页面或对话框。活动页面被指派给由 $.mobile.activePageClass 指定的 CSS 类。

■ $.mobile.firstPage

这是页面容器（$.mobile.pageContainer）内定义的第一个页面。例如，当不存在 location.hash 值，或者是禁用了$.mobile.hashListeningEnabled 时，则会显示$.mobile.fristPage。例如，在一个多页面文档中，默认情况下会最先显示$.mobile.firstPage。

■ $.mobile.pageContainer

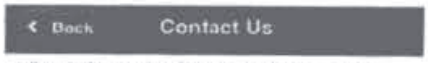
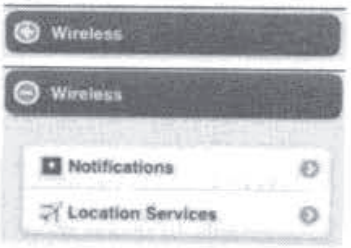
所有页面存在的 HTML 容器。在 jQuery Mobile 内，body 元素是包含所有页面的容器。所有通过 Ajax 载入的页面，以及多文档页面中的所有内部页面都会存在于页面容器内。

8.5 数据属性

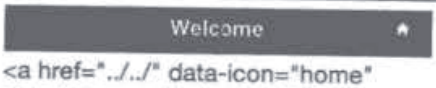
jQuery Mobile 的数据属性提供了通过简单的 HTML 标记来增

序的能力。有关数据属性的完整列表，以及对数据属性的描述和示例，请见表 8-1。该表按照字母顺序排列。

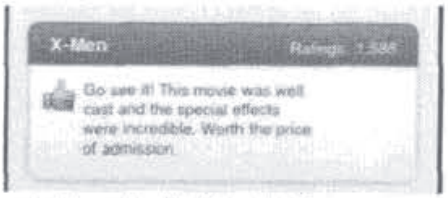
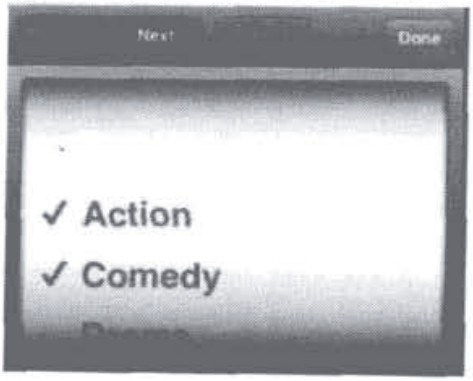
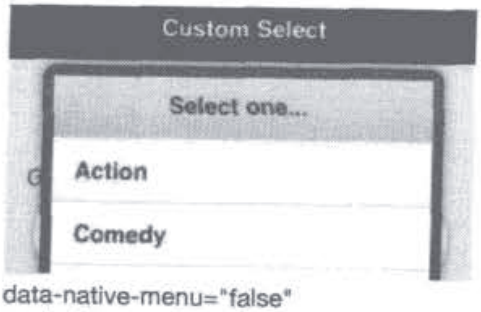
表 8-1 jQuery Mobile 的数据属性

属性	描述	示例
data-ajax	该属性可以附加到链接、按钮或表单上。当该属性为 false 时，它会强制重新载入一个页面（绕过了 Ajax 和页面转换）。例如，如果某个页面是使用 Ajax 打开的，当通过链接在该页面上打开一个多页面文档时，会需要该属性。默认情况下，Ajax 导航是启用的	multi-page
data-add-back-btn	该属性被附加到一个页面容器上。当该属性为 true 时，在页面的页眉上会自动出现一个回退按钮。要想出现回退按钮，则浏览器的访问历史记录中必须有页面存在。在默认情况下，回退按钮是禁用的。	 <div data-role="page" **data-add-back-btn="true"**>
data-back-btn-text	该属性被附加到页面容器上。回退按钮的默认文本是“Back”。通过更新该属性的这个值，可以覆盖该文本	 <div data-role="page" data-add-back-btn="true" **data-back-btn-text="Previous"**>
data-collpased	通过添加该属性，可以配置能够折叠(data-collapsed="true") 和展开(data-collpased="false")的折叠容器。在默认情况下，可折叠的部分以展开的形式显示（见程序清单 6-10）	 Wireless
data-corners	该属性可以附加到链接或按钮上。当该属性为 false 时，jQuery Mobile 框架会移除按钮的圆角。默认情况下，按钮是圆角的。例如，右边显示的“Disagree”按钮已经移除了圆角	 Disagree
data-count-theme	该属性可以为徽章或计数泡设置其他主题（见程序清单 5-9）	 <ul data-role="listview" data-count-theme="e">
data-direction	该属性被附加到链接、按钮或表单。当该属性设置为 reverse 时，它会应用一个反向转换。例如，正向的“滑动”转换是向左滑动。而反向的“滑动”转换则是向右滑动。当转换回历史记录中的页面时，默认应用的是反向转换	Home

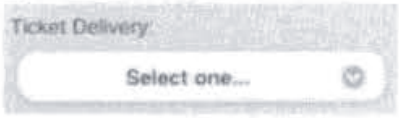
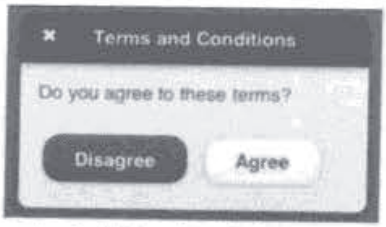
续表

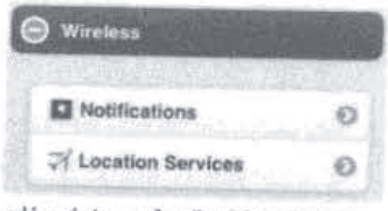
属性	描述	示例
data-divider-theme	设置列表分割线的主题（见程序清单 5-3）	 <li data-role="list-divider" **data-divider-theme="a"**>
data-dom-cache	该属性允许用户在 DOM 内缓存页面。默认情况下，该属性被设置为 false，因此 jQuery Mobile 框架只会在 DOM 内主动缓存“from”页面和“to”页面。建议不要更改该值，以保持 DOM 是轻量级的	
data-filter	该属性被附加到列表中，而且当该属性的值为 true 时，会在列表的上方添加一个搜索栏（见程序清单 5-10）	 <ul data-role="listview" **data-filter="true"**>
data-filter-placeholder	为搜索过滤器设置占位符（提示）文本。默认的占位符文本是“Filer items...”（见程序清单 5-10）	 <ul data-role="listview" data-filter="true" **data-filter-placeholder="Search..."**>
data-filter-theme	为搜索过滤器设置主题	 <ul data-role="listview" data-filter="true" **data-filter-theme="e"**>
data-fullscreen	该属性被附加到页面容器。当该属性设置为 true 时，则内容区域会以全屏模式显示。通常情况下，在观看相片和视频时，会使用该行为（见程序清单 3-1）	<div data-role="page" **data-fullscreen="true"**>
data-icon	该属性被附加到链接或按钮。例如，将该属性的值设置为 home，则会显示来自 jQuery Mobile 图标托盘的 home 图标。有关该属性值的完整列表，请见表 4-1	 Home
data-iconpos	该属性被附加到按钮或链接上。该属性会对图标的位置进行定位，它可以设置为“top”、“bottom”、“left”、“right”或“notext”（见程序清单 6-4）。“notext”会移除图标的默认文本，使按钮上只剩下图标。默认情况下，图标是左对齐的	 Home
data-iconshadow	该属性可以与 data-icon 属性结合使用。当该属性为 false 时，jQuery Mobile 框架会移除按钮图标的阴影。阴影在默认情况下是显示的。例如，右边显示的“加号”图标已经移除了阴影	 Plus

续表

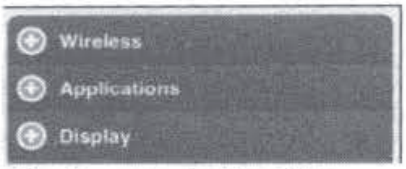
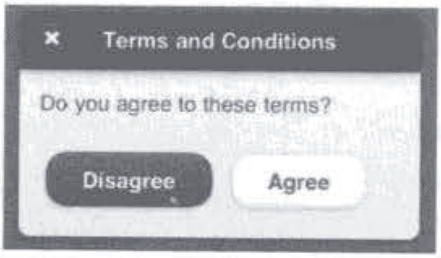
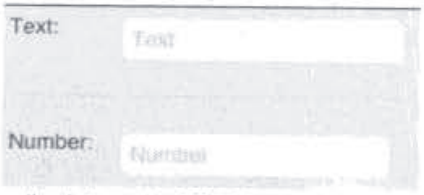
属性	描述	示例
data-id	该属性被附加到页脚，通常与标签栏一起使用。通过为活动页面和目标页面的页脚添加该属性，可以让页脚在页面转换期间停留在适当位置。凡是带有该页脚的页面，为了让页脚停留在适当位置，该属性的值在转换前后的页面上必须相同，而且页眉工具栏和页脚工具栏必须设置为 data-position="fixed"。否则，在页面转换期间则无法看到它们	<div data-role="footer" **data-id="myFooter"** data-position="fixed">
data-inline	该属性被附加到链接或按钮。默认情况下，body 内容中的所有按钮都被样式化为块级元素，而且会填满屏幕的整个宽度。如果你需要一个紧凑的按钮，其宽度与按钮文本和图标的宽度相同，则可以将该属性的值设置为 true。内联的块（block）是以内联的形式放置的（例如，作为相邻的内容放置在同一行），但是它仍然表现出块级的行为。为按钮添加 data-inline 后，按钮会并排显示（见程序清单 4-1）	 <a href="#agree" data-role="button" **data-inline="true"**>
data-inset="true"	该属性被附加到列表上。当该属性设置为 true 时，会对列表条目进行样式化，以便它们在显示时，不会顶着屏幕左右两侧的边缘，而是以圆角形式显示。内嵌的方式有助于从视觉上对列表分割成不同的组	 <ul data-role="listview" **data-inset="true"**>
data-native-menu	默认情况下，选择菜单会为 OS 启动本地的选择选择器。为了以一种自定义的 HTML/CSS 视图的方式呈现选择菜单，可以将该属性的值设置为 false（见程序清单 4-17）	 data-native-menu="true"  data-native-menu="false" <select name="genre" **data-native-n**

续表

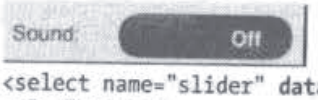
属性	描述	示例
data-placeholder	占位符可以用来为未选中的选择菜单显示提示文本，它要求用户做出选择（见第 4 章的“占位符选项”一节）	 `<option value="" data-placeholder="true">Select one...</option>`
data-position	该属性被附加到页眉或页脚。当该属性被设置为 fixed 时，它会分别将页眉和页脚固定在页面的顶部和底部	`<div data-role="header" data-position="fixed">` `<div data-role="footer" data-position="fixed">`
data-prefectch	当将该属性添加到一个链接或按钮上时，jQuery Mobile 框架会在后台将页面缓慢地载入到 DOM 中。建议构建单独的页面（单页面模板），并使用 data-prefecth 属性预先载入通常会访问到的附属页面。与多页面的策略相比，该策略要更加简单，而且性能更好	`<a href="reviews.html" data-prefetch>Movie Reviews</a>`
data-rel=“back”	该属性被附加到链接或按钮上。当该属性被设置为“back”时，链接会模拟回退按钮，返回到一个历史访问条目（window.history.back()），并且忽略链接默认的 href。对于 C 级浏览器（不支持 JavaScript）来说，它会忽视 data-rel，而且 href 属性用用作一个备选项（fallback）（见程序清单 2-7）	`<a href="home.html" data-role="button" data-rel="back">Disagree</a>`
data-rel=“dialog”	该属性被附加到链接或按钮上。可以将该属性的值设置为 dialog，以表示你希望将目标页面样式化为模态对话框（见程序清单 2-5）	 `<a href="dialog.html" data-role="button" data-rel="dialog" data-transition="slideup">Show Dialog</a>`
data-role=“button”	该属性被附加到链接上。将该属性设置为 button，则可以将链接样式化为按钮的形式	 `<a href="#movies" data-role="button">Show Movies</a>`

data-role=“collapsible”	为了创建一个可以展开和折叠的内容块，需要使用一个包含该属性的元素对该内容块进行包装	
		`<div data-role="collapsible">`
		`<h3>Wireless</h3>`
		`</div>`

续表

属性	描述	示例
data-role="collapsible-set"	可折叠的集合与可折叠的块相似，只不过可折叠集合的组成区域在视觉上是组合在一起的，而且一次只能展开一个区域，这使得可折叠集合的外观很像手风琴（见程序清单 6-11）	 Wireless
data-role="content"	该属性被附加到包含内容主体（content body）的 div 元素上。该元素是可选的（见程序清单 2-1）	
data-role="controlgroup"	如果想把按钮分组到一起，可以将按钮放在一个控件组内（见程序清单 4-8）	 In Theatres Coming Soon
data-role="dialog"	该属性被附加到页面容器上。可以用该属性替换 data-role="page"。将该属性的值设置为 dialog，会以模态对话框的形式来样式化页面	
Data-role="fieldcontain"	该属性被附加到包含表单字段的 div 元素上。当该属性的值为 fieldcontain 时，它会在被包含的字段之间添加行分割线	 <label><input>
data-role="footer"	该属性会创建页脚容器。页脚是可选的（见程序清单 2-1）	

续表

属性	描述	示例
data-role="header"	该属性会创建页眉容器。页眉是可选的（见程序清单 2-1）	
data-role="list-divider"	将该属性添加到列表上，可以对页眉分段（见程序清单 5-3）	 <li data-role="list-divider">Mon
data-role="listview"	该属性用于创建列表视图（见程序清单 5-1）	 <ul data-role="listview"> Action
data-role="navbar"	该属性用来创建导航栏或标签栏。导航栏可以附加到页眉或页脚上（见程序清单 3-8）	 ...
data-role="none"	通过为任意表单或按钮添加该属性，可以使框架不应用任何增强样式或脚本	 <input type="text" name="name" id="name" value="" data-role="none" />
data-role="page"	该属性定义了页面容器（见程序清单 2-1）	 <!-- header --> <!-- content --> <!-- footer -->
data-role="slider"	该属性用来创建一个开关控件（见程序清单 4-25）	 <select name="slider" data-role="slider"> <option value="off">Off</option> <option value="on">On</option> </select>
data-shadow	该属性的值为 false 时，框架会移除按钮的阴影。在默认情况下，阴影是显示的。右边显示的“Disagree”按钮已经移除了阴影	 Disagree

续表

属性	描述	示例
data-split-icon	在构建拆分按钮时，设置附属按钮的图标（见程序清单 5-6）	 <ul data-role="listview" **data-split-icon="star"**>
data-split-theme	在构建拆分按钮时，设置附属按钮的主题	 <ul data-role="listview" data-split-icon="delete" **data-split-theme="v"**>
data-theme	将该属性添加到所有的容器和页面组件中，创建可主体化的设计。有关可用主题的完整列表，请见第 7 章	<div data-role="header" **data-theme="b"**>
data-title	该属性被附加到页面容器，并设置页面的标题（见第 2 章的“设置内部页面的页面标题”一节）	<div data-role="page" **data-title="Welcome"**>
data-transition	该属性可以附加到链接、按钮和表单上。当页面之间发生转换时，该属性的值可以用来设置页面转换所使用的 CSS 转换效果。有关可用转换的完整列表，请见表 2-1	Flip
data-type=“horizontal”	默认情况下，按钮是垂直放置的。通过添加 data-type=“horizontal”属性，可以以水平方式样式化按钮	 <div data-role="controlgroup" **data-type="horizontal"**> In Theatres Coming Soon
data-url	该属性被附加到页面容器上。该属性的值是页面唯一的定位符，将会在浏览器的 URL 地址栏显示。默认情况下，jQuery Mobile 会基于页面的 URL 来指派唯一的定位符。但是，如果希望改变 URL，则可以将期望的 URL 设置为 data-url 属性的值。例如，在重定向之后，你可以覆盖 URL。如果你想隐藏文件名，只显示 URL 路径，则是可以更新 data-url 属性，以将文件名排除在外	<div data-role="page" **data-url="/override/url/path/or/filename.html"**>

8.6 总结

在本章，我们讲解了如何配置 jQuery Mobile，并查看了构建动态页面所需要的许多常见的 API 特性。无论你希望以全局方式来更改默认的转换，还是想在所有页面上显示回退按钮，jQuery Mobile 都会允许我们重新配置许多常见的设置。我们还讲解了很多 jQuery Mobile 已经向公众开放的常见的方法、事件和属性。在需要以程序方式更新移动 Web 应用程序时，这些 API 特性会相当有用。最后，我们还讲解了 jQuery Mobile 的所有数据属性。当你需要快节奏地开发 jQuery Mobile 项目时，可以方便地参考这个数据属性列表。在这个属性列表中，每一个属性都包含了一个简短的介绍、示例，而且某些属性还带有示意图。

在下一章，我们会深入讲解如何高效地使用服务。我们会看到如何将 jQuery Mobile 页面与客户端和服务器端的集成解决方案结合起来。

第9章

服务集成策略

在构建 Web 应用程序时，有两种主要的访问策略来载入数据。一种是传统的服务器端的访问策略，另外一种是 Web 2.0 的客户端访问策略。在本章，我们会演示几个将 jQuery Mobile 与这两种访问策略相集成的示例，还会讨论每一种访问策略的优势。jQuery Mobile 很够很好地与这两种策略相集成，因此，你可以选择最符合你的应用程序需要的访问策略。

开始讲解之前，我们会先看两个客户端集成的例子。由于社交媒体的广泛流行，我们的第一个例子会演示如何与 Twitter 的 RESTful API 相集成。RESTful 的 API 是轻量级的 Web 服务，而且通常优于传统的 Web 服务，原因是它们的安装很方便，而且具有灵活的响应类型（JSON、XML）。在 Twitter 示例之后，我们会创建自己的 RESTful API，它允许用户进行注册，以获得免费的电影奖项。这个注册示例有助于演示从 jQuery Mobile 应用程序 POST 到一个 RESTful 的 API 的能力。此外，通过该示例，你还会了解如何在客户端安装 RESTful API。

然后，我们会讲解服务器端的集成策略，并分别实现一个 GET 数据和 POST 数据的使用案例。出于比较的目的，我们会将客户端集成示例以服务器端的方式重新实现。你会惊奇地发现，当使用客户端的（模型-视图-控制器）MVC 访问策略来获取数据时，我们的页面标记会相当清爽。

最后，由于地理定位和地图视图（map view）的流行，我们会讲解如何将 jQuery Mobile 与 HTML5 的地理定位 API 和 Google Maps 相集成。

9.1 使用 RESTful 服务的客户端集成

大多数社交媒体网站都提供了公共的 API，以供用户访问它的数据。Twitter[1]、LinkedIn[2]和 Facebook[3]集成在 Web 上都相当常见，而且对它们来说，RESTful 的集成是最常见的访问策略。在本节，我们会使用两个不同的 RESTful API 与 jQuery Mobile 进行集成，这样我们可以看到客服端访问策略是如何在 jQuery Mobile 内运行的。

9.1.1 使用 Ajax 的客户端 Twitter 集成

在我们的第一个客户端示例中，我们会使用 Twitter 的 RESTful API 与 jQuery Mobile 集成。Twitter 是一种非常流行的社交媒体站点，它允许用户发送或“推特（tweet）”与主题、事件或随机意见相关的简短消息。在我们的电影 app 中，这会很有用，它可以让用户实时搜索 Twitter，以查找可能感兴趣的某一电影的评论。例如，除了查看某一特定电影的其他评论，我们还可以提供一个到 Twitter 的简便链接，以显示与电影相关的最新“推特”。在我们的用户查看页面中，我们在页眉中添加了一个 Twitter 按钮，用户可以单击该按钮来查看与电影相关的最新“推特”（见图 9-1，相关代码见程序清单 9-1）。

图 9-1 Twitter 按钮出现在电影评论页面的页眉中

1 见 http://dev.twitter.com/console.
2 见 http://developer.linedin.com/community/apis.
3 见 http://developers.facebook.com/.

程序清单 9-1 在电影评论页面的页眉中显示 Twitter 按钮（ch9/reviews.html）

```
<div data-role="page" id="reviewsPage">
    <div data-role="header">
        <h1>Reviews</h1>
        <a href="twitter.html" id="twitterBtn" class="ui-btn-right"
                data-icon="custom" data-iconpos="notext"></a>
    </div>
    ...
</div>
```

当用户单击 Twitter 按钮时，我们可以在 Twitter 上搜索与当前电影相关的所有最近“推特”，并将结果载入到我们的 Twitter 结果页面中（见图 9-2，相关代码见程序清单 9-2）。

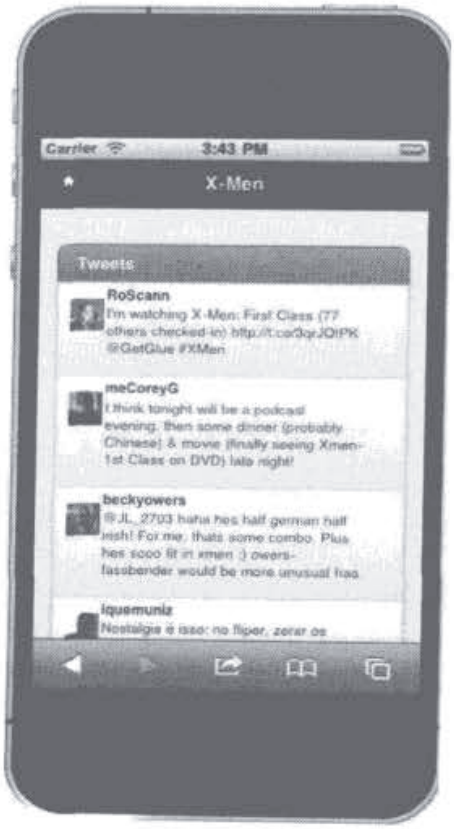


图 9-2 Twitter 结果页面

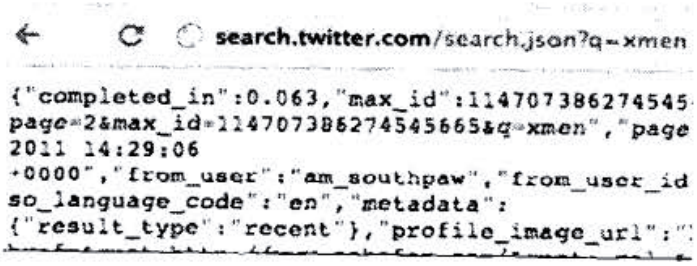
程序清单 9-2 Twitter 结果页面（ch9/twitter.html）

```
<div data-role="page" id="twitterPage">
        ...
        <div data-role="content">
          <ul id="tweet-list" data-role="listview" data-inset="true">
            <li data-role="list-divider">Tweets</p></li>
          </ul>
        </div>
</div>
```

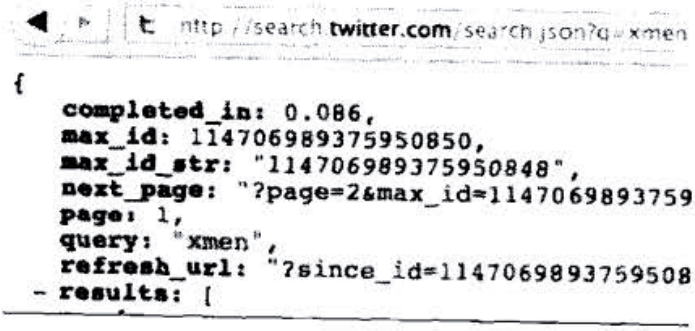
程序清单 9-2 中显示的 tweet-list id 是我们的占位符，我们可以在这里添加 Twitter 的搜索结果。Twitter 的搜索 API 会返回很多数据元素，但是，我们真正感兴趣的是“推特”文本、发布该“推特”的用户，以及用户的头像（profile image）。

提示：要查看 Twitter 的搜索 API 提供的所有可用的数据元素，需要在浏览器中启动字符串地址 http://search.twitter.com/search.json?q=xmen。这是一个基本的搜索 API，其中“q”参数的值是我们的搜索关键词。在该例子中，我们搜索的是带有”xmen”关键词的所有“推特”。

提示：大多数浏览器会以一种未格式化的样式来显示 JSON 响应，这对用户来说相当不方便。



而 Firefox 具有一个 JSON 查看器插件，它以一种更为结构化的格式来显示 JSON 响应。



而且，如果你需要验证 JSON，则可以使用 JSONLint[1]工具。

jQuery Mobile 内置了对 Ajax 的支持，这样就可以在不依赖第三方 JavaScript 框架的情况下，轻松地进行 RESTful 的集成。这一支持是通过 jQuery Mobile 扩展的 jQuery API[2]实

1 见 http://jsonlint.com/.
2 见 http://jquery.com/.

现的。在 jQuery 中，$.ajax[1] API 是用于 RESTful 集成的首选解决方案，这是因为它比较简单，而且还具有灵活的配置选项（超时、缓存等）。

要使用 Twitter 的 RESTful API 来集成 jQuery Mobile，有必要采取如下步骤（见程序清单 9-3）。

1．在单击 Twitter 按钮时，最先显示 jQuery Mobile 活动指示器，以便用户能够看到有活动正在后台执行。

```
$( "#twitterBtn" ).bind( "click", function(e) {
    $.mobile.showPageLoadingMsg();
```

2．接下来，将 Twitter 结果页面载入到当前页面的 DOM 中。如果该页面已经存在于 DOM 中，我们将重新载入和更新被缓存的页面。

```
$.mobile.loadPage("twitter.html", { reloadPage: true });
```

3．在我们的 Twitter 页面被增强之前，我们将发送一个 AJAX 请求到 Twitter API，以收集搜索结果。

```
$("#twitterPage").live("pagebeforecreate", function(){
   $.ajax({...
```

4．我们的 url 选项被配置为 Twitter 的 RESTful API，我们的搜索查询被配置为找到所有包含关键词“xmen”的“推特”。

```
url: "http://search.twitter.com/search.json?q=xmen"
```

5．由于 Twitter API 存在于另外一个域，我们需要将 dataType 选项设置为 jsonp。通常情况下，Web 上不允许跨域进行的通信，但是 JSON[2]提供了一种用于跨域集成的受信方式。

```
dataType: "jsonp"
```

6．最后，实现我们的 success 回调函数，以迭代每一个搜索结果。为每一行创建一个列表，并且在列表容器中添加新的标记。

```
success: function( response ) {...
```

[1] 见 http://api.jquery.com/jQuery.ajax/.
[2] 见 http://en.wikipedia.org/wiki/JSONP.

程序清单 9-3 客户端 Twitter 集成（ch9/twitter.js）

```
$( "#reviewsPage" ).live( "pageinit", function(){
    $( "#twitterBtn" ).bind( "click", function(e) {
        $.mobile.showPageLoadingMsg();

        // Reload Twitter results page even if it's already in the DOM
        $.mobile.loadPage("twitter.html", { reloadPage: true });

        // Prevent default click behavior
        return false;
    });
});

$( #twitterPage" ).live("pagebeforecreate", function(){
   $.ajax({
      url: "http://search.twitter.com/search.json?q=xmen",
      dataType: "jsonp",
      success: function( response ) {
         // Generate a list item for each tweet
         var markup = "";
         $.each(response.results, function(index, result) {
             var $template = $('<div><li><img class="ui-li-icon profile">
<p class="from"></p><p class="tweet"></p></li></div>');
             $template.find(".from").append(result.from_user);
             $template.find(".tweet").append(result.text);
             $template.find(".profile")
                          .attr("src", result.profile_image_url);
             markup += $template.html();
         });

         // Append the Tweet items into our Tweet list and refresh the
         // entire list.
         $( "#tweet-list" ).append(markup).listview( "refresh", true );

         // Transition to the Twitter results page.
         $.mobile.changePage( $("#twitterPage") );
      },
   });
});
```

在这个例子中，当用户单击 Twitter 按钮时，我们选择按需载入 Twitter 结果。你也可以预先获取（pre-fetch）Twitter 数据，以便在用户单击按钮时，能立刻看到 Twitter 结果。要设置该策略，需要在 Twitter 按钮上添加 data-prefetch 属性：

```
<a href="twitter.html" id="twitterBtn" class="ui-btn-right" data-icon="custom" data-
iconpos="notext" data-prefetch></a>
```

现在，通过按钮的默认单击行为就可以处理页面的更改，这样，在 Twitter 结果添加到列表之后，我们就可以移除为按钮自定义的单击处理程序和$.mobile.changePage()调用。

在产品（production）使用案例中，我们可以配置$.ajax 方法的 timeout 和 error 回

调，以处理任何没有响应或者是不可用的 API。例如，如果 Twitter API 没有响应，则将该情况通知用户。

```
timeout: 6000, // Timeout after 6 seconds
error: function(jqXHR, textStatus, errorThrown) {
    $.mobile.hidePageLoadingMsg();

    // Show error message
    $( "<div class='ui-loader ui-overlay-shadow ui-body-e
     ui-corner-all'><h1>"+ $.mobile.pageLoadErrorMessage +"</h1></div>" )
        .css({ "display": "block", "opacity": 0.96, "top": 100 })
        .appendTo( $.mobile.pageContainer )
        .delay( 800 )
        .fadeOut( 1000, function() {
            $( this ).remove();
        });
}
```

9.1.2 使用 Ajax 的客户端表单 POST

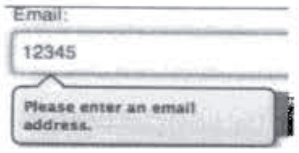
前面的例子是一个向 Twitter API 发送 GET 请求的使用案例。当读取一个 API 时，经常会用到 GET 请求，而且在没有指定类型的情况下，$.ajax 方法默认使用的就是这个类型。在我们的下一个例子中，我们会创建自己的 RESTful API，它允许用户发送 POST 请求。我们创建一个 API，以便用户为了赢得电影奖项进行注册。我们的用户界面包含一个只需要邮件地址的简单表格（见图 9-3，相关代码见程序清单 9-4）。

程序清单 9-4 使用 Ajax 的客户端 POST 注册表单（ch9/register-client.html）

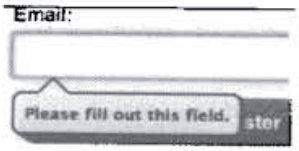
```
<div data-role="page" id="registrationPage" data-theme="d">
    <div data-role="header">
      <h1>Registration</h1>
    </div>
    <div data-role="content">
      <form id="register" method="post">
        <label for="email">Email:</label>
        <input type="email" name="email" id="email" placeholder="Email"
required />
        <input type="submit" id="submit" value="Register" />
      </form>
    </div>
</div>
```

图 9-3 使用 Ajax 的客户端 POST 注册表单

提示：邮件地址的输入字段包含 3 个 HTML5 新属性。type="email" 字段提供了两个好处。第一，当该字段获得焦点时，屏幕中会显示一个 QWERTY 键盘，而且该键盘会显示几个有用的电子邮件按键（见图 9-3）。第二，在提交表单时，它会验证字段中包含的邮件地址是否有效。例如，在较新的桌面浏览器中，如果用户输入了一个无效的邮件地址，则会提示如下信息：



此外，当用户提交表单时，required 属性会声明邮件字段不能为空。如果邮件字段为空，用户会看到如下警告信息：



最后，placeholder 属性会为输入字段添加提示文本。尽管这些特性很有用，但是并非当今所有的浏览器都对其提供支持。Peter-Paul Koch 有一个站点[1]显示了所有可用的输入属性，以及相关的浏览器支持。

[1] 见 http://www.quirksmode.org/html5/inputs_mobile.html 和 http://www.quirksmode.org/html5/input.html

为了在客户端处理表单提交，我们需要为提交按钮附加一个事件监听器。要向我们的 RESTful API 提交一个 POST 请求，有必要采取如下步骤（见程序清单 9-5）。

1. 首先，我们需要截获并覆盖默认的提交行为。现在，我们准备通过 RESTful API 来提交表单。

```
$("form").submit(function () {
```

2. 其次，我们需要显示 jQuery Mobile 活动指示器，以便用户能够看到有活动正在后台执行。

```
$.mobile.showPageLoadingMsg()
```

3. 接下来，使用所有必要的选项来设置$.ajax 请求。

- 我们的 url 选项被配置为在本地设置的 RESTful 的新资源，以处理客户端的注册：

```
url: "http://localhost:8080/jqm-webapp/rest/register"
```

稍后我们会讲解 RESTful 的实现。

- 接下来，将 type 选项设置为 POST。在创建新的条目时，建议使用 POST 类型，而且它要比 GET 更安全，这是因为它不会将数据属性用作 URL 上的查询字符串参数：

```
type: "POST"
```

- 再次将 dataType 选项设置为 jsonp，这是因为我们的 RESTful API 也可以运行在单个域中。

```
dataType: "jsonp"
```

- jsonp 选项定义了会处理响应的回调函数：

```
jsonp: "jsoncallback"
```

处理 jsonp 请求的任何 RESTful 的资源必须产生一个 JavaScript 响应。该响应实际上是包装在 JavaScript 函数内的 JSON 数据。例如，我们的 RESTful API 的响应会包含一个成功注册的邮件地址，它包装在一个回调函数中。该回调函数的名字需要被设置为 jsonp 选项的值：

```
jsoncallback({"email":"BradPitt@gmail.com"})
```

- 数据选项包含需要发送给 RESTful 的资源的数据。在该例中，我们会发送所有的表单值和 URL，并使用 jQuery 的 serialize 方法对它们进行编码。

```
data: $("form#register").serialize(),
```

- 最后一个选项是 success 处理程序。在从 RESTful API 接收到一个成功的响应之

该程序会得以处理。在我们的例子中，我们将用户转到一个 Thank You 页面，并将成功注册的邮件地址作为数据参数进行传递，以进行确认。

```
success: function( response ) {$.mobile.changePage( "register-
thanks.html", {
        data: {"email": response.email}} );
    }
```

程序清单 9-5 使用 Ajax 的客户端 POST（ch9/register.js）

```
$("#registrationPage").live("pageinit", function(){
  $("form").submit(function () {
    $.mobile.showPageLoadingMsg();

    $.ajax({
        url: "http://localhost:8080/jqm-webapp/rest/register",
        type: "POST",
        dataType: "jsonp",
        jsonp: "jsoncallback",
        data: $("form#register").serialize(),
        success: function( response ) {
           $.mobile.changePage( "register-thanks.html",
                { data: {"email": response.email}} );
        }

        return false; // Prevent a form submit
    });
});
```

在成功注册之后，用户会被转到一个 Thank You 页面，我们会通过这个页面向用户显示：他们赢得了什么奖项，以及奖项发送到了哪个邮件地址（见图 9-4，相关代码见程序清单 9-6）。

程序清单 9-6 在使用 Ajax 的客户端 POST 之后显示 Thank You 页面(ch9/register-thanks.html)

```
<div data-role="page" id="thanksPage" data-theme="d">
  <div data-role="header">
    <a href="/ch9/register-client.html" data-icon="home"  data-iconpos="notext" data-
direction="reverse"></a>
        <h1>Thanks</h1>
  </div>

  <div data-role="content" class="thanks">
    <p>Thanks for registering.  One FREE movie pass was just sent to:  <span
class="email"></span></p>
    <img src="images/free-movie.jpg">
  </div>

</div>
```

图 9-4 在使用 Ajax 的客户端 POST 之后，显示的 Thank You 页面

> 提示：在为网站设计导航策略时，注意为用户提供一些导航选项，以避免用户进入死胡同。在 jQuery Moblie 中，一个简单的解决方案是在页眉栏显示 home 按钮，并使用反向链接将其重定向到主页面。

```
<a href="home.html" data-icon="home" data-iconpos="notext" data-
direction="reverse"></a>
```

在执行 changePage 方法调用时，也可以将邮件地址作为一个数据属性传递到 Thank You 页面。这个数据属性会添加到页面的 data-url 属性后面：

```
data-url="/ch9/register-thanks.html?email=BradPitt%40gmail.com"
```

在增强 Thank You 页面之前，我们要先获取上面提到的这个邮件地址，然后将其绑定到 Thank You 页面中的邮件占位符上（见程序清单 9-7）。

程序清单 9-7 在 Thank You 页面添加邮件地址（ch9/register.js）

```
$("#thanksPage").live("pagebeforecreate", function(){
    var email = getParameterByName("email", $(this).jqmData("url"));
```

```
        $(".email").append(email);
    });

    function getParameterByName(param, url) {
        var match = RegExp('[?&]' + param + '=([^&]*)').exec(url);
        return match && decodeURIComponent(match[1].replace(/\+/g, ' '));
    }
```

在服务器端的 RESTful 实现（用来处理注册）是使用 Jersey[1]实施的，并且是部署在 Tomcat[2]上的（见程序清单 9-8）。

程序清单 9-8 用来处理注册的 RESTful 的资源(com.bmb.jqm.resource.RegisterResourse.java)

```
@Path("/register")
public class RegisterResource {

    @Produces("application/x-javascript")
    public Response register(@QueryParam("jsoncallback")
        @DefaultValue("jsoncallback") String callback,
        @QueryParam("email") String email) {
        Registration registration = new Registration();
        registration.setEmail(email);
        // Save registration...

        // Return registration in response as jsonp
        return Response.status(Response.Status.OK).entity(new
        JSONWithPadding(registration, callback)).build();
    }
}
```

通过分析该资源，我们来了解一下 Jersey 的注解（annotation）。

- @Path 注解定义了资源负责处理的路径。在本例中，RegisterResource 对象会处理所有发送到“*/rest/register”的请求。Jersey 在 web.xml 文件中配置，其中有两个需要安装的配置选项（见程序清单 9-9）。首先，我们需要定义用来部署所有资源的包（package），然后，我们需要定义应该通过 Jersey 容器发送的 URL 模式。我们已经确定，所有的 RESTful 的资源都在“com.bmb.jqm.resource”包中声明，而且我们会通过 Jersey 容器来为带有“/rest/*”的所有 URL 指定路径。

- @Produces 注解定义了响应的 MIME 类型。当有域需要资源，以返回一个 JavaScrit 响应时，我们选择对这些域公开 RSETful API。这可以让客户端通过 jsonp 请求来访问 API。

[1] 见 http://jersey.java.net/.
[2] 见 http://tomcat.apache.org/.

■ register 方法接受两个输入参数。第一个是回调函数的名字。客户端可以发送回调函数的名字，但这不是必需的。如果没有提供其他名字，服务器会将回调函数的名字默认为“jsoncallback”。第二个参数是用户用来注册的邮件地址。

■ 服务器现在可以处理用户的注册过程，并生成一个响应。在这个例子中，我们会返回一个响应，该响应中包含转换为 JSON 并包装在回调函数内的注册对象。

```
jsoncallback({"email":"BradPitt@gmail.com"})
```

程序清单 9-9 Jersey 配置（web.xm）

```
<servlet>
  <servlet-name>Jersey REST Service</servlet-name>
  <servlet-class>
    com.sun.jersey.spi.container.servlet.ServletContainer
  </servlet-class>
  <init-param>
      <param-name>com.sun.jersey.config.property.packages</param-name>
      <param-value>com.bmb.jqm.resource</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
      <servlet-name>Jersey REST Service</servlet-name>
      <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
```

可以看到，jQuery Mobile 能够与 RESTful API 很好地集成。无论我们需要读取数据还是提交数据，内置的 jQuery 库都提供了所有函数，通过这些函数，我们可以在客户端上方便地管理 RSTful 的生命周期。

9.2 使用 MVC 的服务器端集成

在本节，我们将重点关注服务器端的访问策略。在 Web 上，一种常见的策略是与模型-视图-控制器（MVC）框架相集成。我们会讲解其样式与客户端示例相似的两个 MVC 示例。在第一个例子中，我们将客户端注册使用案例通过服务器端来实现。通过该例子我们会看到，在 jQuery Mobile 中，通过客户端访问策略实现的使用案例与通过服务器端访问策略实现的使用案例相同。在第二个例子中，我们会讲解如何实现一个从服务器 GET 数据的使用案例。

9.2.1 使用 MVC 的服务器端表单 POST

出于比较的目的，我们来看一下如何在服务器端实现我们的注册使用案例。我们

会用到一个注册表单，以允许用户选择接收打折的电影票或免费的电影票（见图 9-5）。

图 9-5 带有 MVC 的客户端 POST 注册表单

这个注册页面中使用的页面标记与采用客户端方式实现的注册页面的标记相似，只不过我们不会覆盖表单提交过程而已。在我们的服务器端注册示例中，当用户单击 Register 按钮时，将会触发我们设定的动作（见程序列表 9-10）。

程序清单 9-10 用于服务器端集成的注册表单（/webapp/ch9/register-sever.html）

```
<div data-role="page" id="registrationPage" data-theme="d">
    <div data-role="header">
        <h1>Register</h1>
    </div>

    <div data-role="content">
     <form id="register" action="/jqm-webapp/mvc/register" method="post">
         <label for="email">Email:</label>
         <input type="email" name="email" id="email" placeholder="Email"
        required />

         <input type="submit" value="Register" data-theme="b"/>
     </form>
    </div>
</div>
```

通过将这个服务器端的注册页面示例（register-server.html）与客户端的注册页面示例（register-client.html）相比较，大家发现有什么明显的区别么？最明显的区别是服务器端的注册页面示例不需要自定义的 JavaScript。因此，相对应的标记页面清爽了好多。

当提交表单时，一个 POST 请求会发送到动作中定义的路径（/jqm-webapp/mvc/register）。部署在 Tomcat 上的 Spring MVC[1]控制器会在服务器端处理该请求。在 web.xml 文件中，我们配置了 servlet 映射，以便所有的“/mvc/*” URL 能够通过 Spring MVC 的分发器(dispatcher servlet)按指定路线发送出去（见程序清单 9-11）。

程序清单 9-11 配置 Spring MVC servlet 映射（/WEB-INF/web.xml）

```
<servlet>
    <servlet-name>jqm-webapp</servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>jqm-webapp</servlet-name>
    <url-pattern>/mvc/*</url-pattern>
</servlet-mapping>
```

从用户界面的角度看，服务器端注册的工作流与客户端的注册相同。该表单被提交、处理，然后显示 Thank You 页面。处理请求，并将请求重定向到 Thank You 页面的控制器代码见程序清单 9-12。

程序清单9-12 Spring MVC 注册控制器(com.bmb.jqm.controller.RegisterController.java)

```
@Controller()
public class RegisterController {

    @RequestMapping(method = RequestMethod.POST)
    public String enroll(@RequestParam("email") String email, HttpSession
        session) {
        // Save registration...

        session.setAttribute("email", email);
        return "redirect:/mvc/register/thanks";
    }
```

[1] 见 http://static.springsource.org/spring/docs/current/spring-framework-reference/html/mvc

```
    @RequestMapping(method = RequestMethod.GET)
    public String thanks() {
        return "register-thanks";
    }
}
```

通过分析控制器，我们来了解一下 Spring MVC 的注解。

- @Controller 注解将类定义为能处理请求的控制器。Spring MVC 使用路径进行了配置，以对名称映射进行分类。例如，所有的“/register/”请求会被分发到 RegisterController。该配置安装在 Spring MVC 的分发器中（见程序清单 9-13）。

- @RequestMapping 注解定义了处理 POST 和 GET 请求的方法。当表单在提交时，一个 POST 请求会发送到 enroll 方法。thanks 方法会处理所有的 GET 请求。例如，在表单被处理之后，我们会重定向到 Thank You 页面，而且在刷新 Thank You 页面时，会触发 thanks 方法。

- @RequestParam 注解会将通过表单提交的邮件地址绑定到邮件输入参数上。在调用 enroll 方法时，我们保存该注册，并将邮件地址放入到会话中，然后重定向到 Thank You 页面（/js/register-thanks.jsp）。

程序清单 9-13 通向控制器映射配置的 Spring MVC 路径（/WEB-INF/jqm-webapp-servlet.xml）

```
<!-- Enable controller mapping by convention. For example: /foo/* will map to
FooController() -->
<bean
class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping" />
```

Thank You 页面的外观看起来与客服端示例中的相同（见图 9-6）。

两者唯一的区别是该页面的生成方式。在服务器端，该页面是作为 JSP 生成的，而且邮件地址与 JSTL 表达式语法绑定到一起（见程序清单 9-14）。由于没有必要使用 JavaScript 来动态生成该页面，因此与客户端示例中动态生成 Thank You 页面相比，该页面的标记也清爽了好多。

程序清单 9-14 服务器端注册之后的 Thank You 页面（/jsp/register-thanks.jsp）

```
<div data-role="page" id="thanksPage" data-theme="d">
    <div data-role="header">
        <h1>Thank You</h1>
    </div>

    <div data-role="content" class="thanks">
        <p>Thanks for registering.  One FREE movie pass was just sent to.
```

```
            <span class="email">${email}</span></p>
            <img src="images/free-movie.jpg">
        </div>
    </div>
```

图 9-6 服务器端的 Thank You 页面

在提交表单之后需要重点考虑的一点是，jQuery Mobile 管理出现在浏览器地址栏中的 URL。例如，在服务器重定向到“/mvc/register/thanks”之后，浏览器的 URL 仍然显示为我们的行动路径（“/jqm-webapp/mvc/register”）。如果没有为该路径实现 GET 请求处理程序，对 Thank You 页面的刷新会产生一个 404 错误。你可以使用下面两个方法来处理这个问题。

- 最简单的解决方法是在控制器上为每一个动作路径实现一个 GET 请求处理程序。我们的 RegisterController#thanks 方法会处理 GET 请求，而且会简单地刷新 Thank You 页面，并重新显示存储在会话中的邮件地址（见程序清单 9-12）。此外，当在 Web 上提交表单时，建议使用 POST，然后再进行重定向，以避免任何二次提交的问题。

- 另外一个方法是，你在页面容器上手动设置 data-url 属性。data-url 属性的值会

显示在浏览器的地址栏中。当开发人员在构造语义路径时，这会给他们提供更多的灵活性。

```
data-url="/manually/set/url/path/"
```

该策略也可以用来隐藏文件名。例如，如果你转向了“/my/movies/index.html”，你可以将页面的 data-url 属性升级为“/my/movies/”，这样就隐藏了 index.html 部分。

9.2.2 使用 MVC 的服务器端数据访问

在前一个例子中，我们看到了如何将表单数据 POST 到服务器。在下面这个例子中，我们仍然使用 GET 请求从服务器获取数据。该例子会从服务器获得一个电影列表，并在一个 jQuery Mobile JSP 页面内显示结果（见图 9-7）。

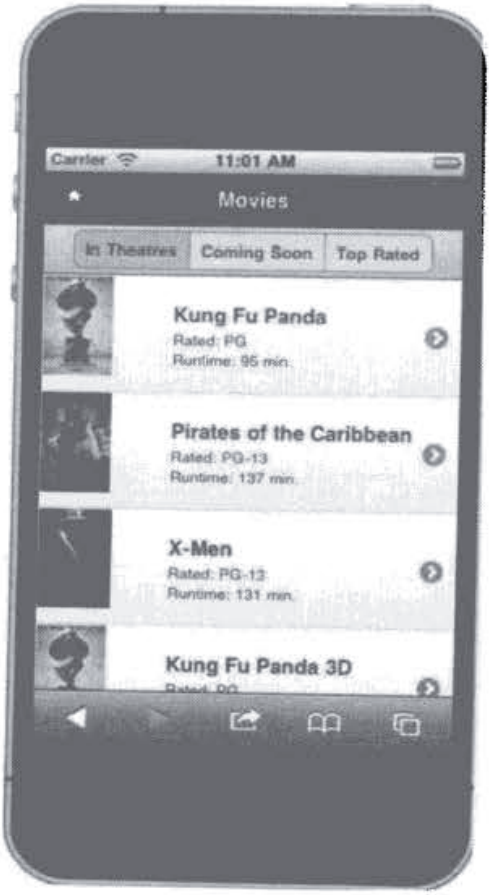


图 9-7 通过服务器端 MVC 访问获得的电影信息

在服务器端，我们安装了一个 Spring MVC 控制器，以处理下面 href 上的 GET 请求：

```
<a href="/jqm-webapp/mvc/movies" data-role="button">Movies</a>
```

当按钮被单击时，将触发一个 GET 请求并将其发送到我们的 MoviesController。MoviesController 会获取我们的电影数据并将响应转发到 JSP 电影页面（见程序清单 9-15）。

程序清单9-15 GET电影数据的MVC控制器(com.bmb.jqm.MoviesController.java)

```
@Controller()
public class MoviesController {

    @RequestMapping(method = RequestMethod.GET)
    public String getMovies(ModelMap model) {
        model.addAttribute("movies", getMovieData());
        return "movies";
    }
}
```

该响应会被转发到电影页面，然后 JSP 在该页面上迭代电影列表，然后为每个结果显示一个单独的列表条目（见程序清单 9-16）。

程序清单 9-16 显示电影数据的 JSP（/jsp/movies.jsp）

```
<div data-role="content">
    <ul data-role="listview">
        <c:forEach var="movie" items="${movies}">
        <li>
            <a href="#">
                <img src="../images/${movie.image}" />
                <h3>${movie.title}</h3>
                <p>Rated: ${movie.rating}</p>
                <p>Runtime: ${movie.runtime} min.</p>
            </a>
        </li>
        </c:forEach>
    </ul>
</div>
```

这种服务器端解决方案的一个优势是，页面标记相当简洁。它不会使用 JavaScript 动态生成页面，无需拼接字符串，也不需要与 jQuery 选择器绑定到一起的动态字段（这会在前面的客户端示例中看到）。

9.2.3 服务器端与客户端的对比

要实施哪种服务访问策略取决于多个因素。如果你已经构建了 Web 应用程序，并且也在 Web 上建立了访问数据所使用的模式，那么出于一致性考虑，你可能希望延续该策略。幸运的是，当实施正确时，jQuery Mobile 能够很好的与任何策略相集成。这

样你就可以有针对性地选择最适合你的特定应用程序需求的模式。下面是每一种实施任何一种策略类型时，需要考虑的事项。

客户端集成：

■ 更快的响应时间。客户端的集成会产生更短的响应时间，原因是他们对点对点服务器的依赖更少。例如，我们可以在服务器端收集 Twitter 数据，但是由于需要与服务器进行通信，因此响应时间会延长。

> **警告：** 尽管客户端集成能提供更短的响应时间，但是在与第三方的 API 进行集成时，要谨慎使用客户端集成，这是因为在服务器端对 API 进行封装时，会更具优势，这样能够更好地隔离你的页面，以防止第三方的修改。例如，Facebook 的 RESTful 的 API 在过去经常会改变，但是现在这种做法实际上已经过时（deprecated）。

■ 实现起来更快。我们在客户端的 Twitter 示例实现非常迅速，这是因为只需要修改客户端的标记。在服务器端实现该任务时，则需要修改客户端和服务器端的组件。

服务器端集成：

■ 更可靠。与客户端的解决方案相比，服务器端的解决方案更可靠，因为你无须关注客户端 JavaScript 的兼容性。

■ 更安全。在实现客户端解决方案时，必须谨慎对待 API 和公开的数据类型。如果使用会公开个人身份信息（Personally Identifiable Information，PII）、个人健康信息（Personal Health Information，PHI）或支付卡行业（Payment Card Industry，PCI）信息的 API 来集成，则客户端的解决方案是不可取的。

■ 页面标记更整洁。在比较使用服务器端访问策略和客户端访问策略实现的页面时，我们已经看到了该情况。使用服务器端访问策略实现的页面与自定义的 JavaScript 没有依赖关系。

■ 组件的单元测试更简单。我一直对这样一个理论深信不疑，即服务器端的单元测试要比客户端的单元测试更简单。然而，在参与过多个 jQuery Mobile 项目的 QUnit 测试之后，我开始觉得客户端的单元测试确实会更成功、更可靠。

9.3 Google Maps 集成

在最近的移动 Web 调查中，大约有 75%的 Web 开发人员使用地理定位，这也使得地理定位成为最受欢迎的 HTML5 API[1]。在构建能够感知位置的应用程序时，通常会使用地图视图来显示感兴趣的点或方位。在 Web 上，通过将地理定位（Geolocation）[2]与 Google Maps[3]结合，能够为构建地图功能提供非常有用的 API。在本节，我们会讲解 jQuery Mobile 是如何与地理定位和 Google Maps 集成的。在开始之前，我们先创建一个能够在地图上标识出你当前位置的例子（见图 9-8）。

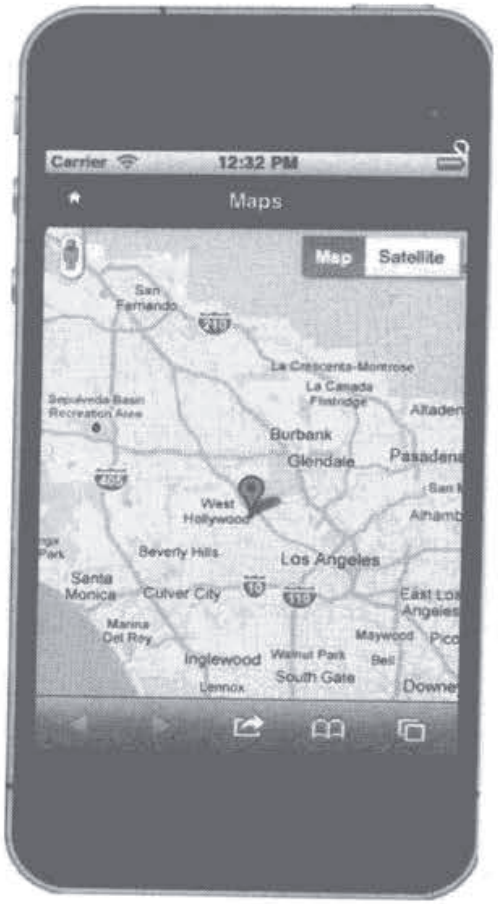


图 9-8 Google Maps 与 jQuery Mobile 的集成

我们的页面只需要最少的标记，因为我们只需要为地图创建内容容器即可（见程序清单 9-17）。

1 见 http://www.webdirections.org/sotmv2011/.
2 见 http://dev.w3.org/geo/api/spec-source.html.
3 见 http://code.google.com/apis/maps/documentation/javasrcipt/basics.html.

程序清单 9-17 用于 Google Maps 集成的 jQuery Mobile 页面（ch9/maps.html）

```
<!DOCTYPE html>
<html>
   <head>
   <meta charset="utf-8">
   <title>Google Maps</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" type="text/css" href="jquery.mobile.css" />
   <style>
     #map-page, #map-canvas { width: 100%; height: 100%; padding: 0; }
   </style>
   <script type="text/javascript" src="jquery.js"></script>
   <script type="text/javascript" src="maps.js"></script>
   <script type="text/javascript" src="jquery.mobile.js"></script>
   <script src="http://maps.google.com/maps/api/jssensor=false"></script>
</head>
<body>
<div data-role="page" id="map-page">
    <div data-role="header">
        <h1>Maps</h1>
    </div>

    <div data-role="content" id="map-canvas">
        <!-- map loads here... -->
    </div>
</div>

</body>
</html>
```

在与 Google Maps 集成时，有必要添加一些内容。

1. 首先，我们需要样式化我们的页面和地图容器，以便它能够全屏显示：

```
#map-page, #map-canvas { width: 100%; height: 100%; padding: 0; }
```

2. 接下来，导入自定义的 JavaScript，用以确定用户的地理位置，并绘制出地图视图。我们稍后会查看该文件的细节。

```
<script type="text/javascript" src="maps.js"></script>
```

3. 然后导入 Google Maps API。

```
<script src="http://maps.google.com/maps/api/jssensor=false">
```

4. 最后，标识我们的地图容器。我们的地图会绘制在该元素内：

```
<div data-role="content" id="map-canvas">
```

自定义的 JavaScript 用来确定用户的地理位置，以及绘制出地图视图，如程序清单 9-18 所示。

程序清单 9-18 用于 Google Maps 集成的 JavaScript（ch9/maps.js）

```
$( "#map-page" ).live( "pageinit", function() {

    // Default to Hollywood, CA when no geolocation support
    var defaultLatLng = new google.maps.LatLng(34.0983425, -118.3267434);

    if ( navigator.geolocation ) {
        function success(pos) {
            // Location found, show coordinates on map
            drawMap(new google.maps.LatLng(
               pos.coords.latitude, pos.coords.longitude));
        }

        function fail() {
            drawMap(defaultLatLng);  // Show default map
        }

        // Find users current position
        navigator.geolocation.getCurrentPosition(success, fail,
            {enableHighAccuracy:true, timeout: 6000, maximumAge: 500000});
    } else {
        drawMap(defaultLatLng); // No geolocation support
    }

    function drawMap(latlng) {
        var myOptions = {
            zoom: 10,
            center: latlng,
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };

        var map = new google.maps.Map(
            document.getElementById("map-canvas"), myOptions);

        // Add an overlay to the map of current lat/lng
        var marker = new google.maps.Marker({
            position: latlng,
            map: map,
            title: "Greetings!"
        });
    }
});
```

当地图页面处于就绪状态时，我们会执行如下步骤，绘制出地图视图。

1. 首先，确定浏览器是否支持地理定位：

```
if ( navigator.geolocation ) {
```

2. 如果浏览器支持地理定位，我们然后尝试获取用户的当前位置：

```
navigator.geolocation.getCurrentPosition(success, fail,
{enableHighAccuracy:true, timeout: 6000, maximumAge: 500000});
```

getCurrentPosition API 接收 3 个参数。第一个参数是 success 回调。这是唯一需要的参数。第二个参数是 error 回调，最后一个参数是可配置的选项。我们已经配置了地理定位查询（lookup），使其使用较高的精度。这会尝试使用 GPS 来进行定位（如果支持的话）。我们还将超时时间配置为 6 秒。如果 6 秒之后还没有找到用户的位置，则会调用 error 回调。最后，我们对这个查询进行了配置，使其能够使用 5 分钟之内的缓存位置。

3. 当成功定位之后，将会调用 success 回调。在本例中，我们使用位置坐标来绘制地图：

```
function success(pos) {
    drawMap(new google.maps.LatLng(
        pos.coords.latitude, pos.coords.longitude))
    }
```

4. 最后，用 drawMap 方法绘制出 Google 地图，而且在地图中心位置显示一个覆盖图标（overlay icon），该图标所在的位置也就是你的位置。然而，不过不支持地理定位或者是定位不成功，则 Hollywood，CA 会作为默认的位置进行显示（见程序清单 9-18）

9.4 总结

在本章中，我们讲解了如何将 jQuery Mobile 与客户端和服务器端数据访问策略相集成。jQuery Mobile 能够很好地与这两种策略相集成，这样用户就可以根据应用程序的需要来选择最合适的数据访问方法。尽管客户端访问策略能提供更好的性能，而且实现起来也更迅速，但是与服务器端的访问策略相比，其可靠性、安全性和可维护性要差一些。

最后，我们讲解了如何将 jQuery Mobile 与地理定位和 Google Maps 相集成的例子。通过使用这两个地图 API，就可以为我们的 jQuery Mobile 应用程序添加地图视图。

在第 10 章，我们会讲解如何将现有的 jQuery Mobile 应用程序与 PhoneGap 结合，并通过 PhoneGap 进行本地分发。

第 10 章

使用 PhoneGap 轻松部署 jQuery Mobile 应用程序

与移动 Web 应用程序相比，本地应用程序具有两个明显的优势。第一，本地应用程序能够通过 app store 分发，最有名的 app store 包括 Apple 的 App Store、Android Market、HP App Catalog、BlackBerry App World 和 Windows Marketplace。当消费者需要搜索、购买、安装本地应用程序，或者需要对本地应用程序进行打分时，App Store 能够简化消费者的用户体验。第二，本地应用程序能够与设备 API 进行交互。例如，本地应用程序能够与大多数设备 API 通信，其中包括通讯录、日历、摄像头，以及网络 API 等。

在本章，我们会讲解如何突破移动 Web 的这些障碍。尤其是，我们会介绍 PhoneGap，以及 PhoneGap 如何为我们的 jQuery Mobile 应用程序架起一座逾越这些障碍的桥梁。我们会以一个现有的 jQuery Mobile 应用程序作为例子，使用 PhoneGap 对其进行包装，然后再部署到本地的 iOS 和 Android 平台上。

我们还会讲解如何在不借助 PhoneGap 的情况下，将 jQuery Mobile 应用程序发布到 app store 中。例如，Open App Market 是一个针对 HTML5 移动 app 的 app store。如果人们觉得本地 app store 的发布过程相当麻烦而且缓慢的话，则可以考虑使用该 app store。

最后，我们会讲解 W3C 在客户端设备 API 方面的进展，浏览器总有一天会支持这

些设备 API。这对移动 Web 而言非常重要，因为它能够在无需依赖外部框架的情况下，允许我们的 Web 应用程序访问设备 API（日历、通讯录、摄像头等）。

10.1 什么是 PhoneGap

PhoneGap[1]是一个开源的开发框架，它允许用户通过使用像 jQuery Mobile 这样的 Web 技术来开发跨平台的本地应用程序。例如，我们可以使用 PhoneGap 来包装一个现有的 jQuery Mobile Web 应用程序，然后将其发布到 PhoneGap 支持的所有本地平台。当前，PhoneGap 支持本地的 iOS、Android、BlackBerry、WebOS 和 Symbian 平台。PhoneGap 除了具有本地发布能力之外，它还公开了一个 API，通过该 API，我们的移动 Web 应用程序能够与设备特定的 API 进行交互，其中包括文件系统、通知（notification）和摄像头等。有关完整列表，请见 PhoneGap 针对不同平台所支持的特性[2]。PhoneGap 允许我们以多种方式来扩展 jQuery Mobile 应用程序，而在此之前我们只能使用本地 SDK 来扩展。

10.2 将 jQuery Mobile 作为一个 iOS app 来运行

在本节，我们将使用 PhoneGap 来包装一个 jQuery Mobile app 应用程序，然后在本地的 iOS 平台上运行。要为 iOS 平台设置 PhoneGap，我们可以参考 PhoneGap 的“Getting Started Guide with iOS”[3]。PhoneGap 具有步骤式安装向导，而且这些安装向导非常详细，甚至具有屏幕截图，因此用户可通过它在任何平台上安装 PhoneGap[4]。要针对 iOS 平台进行开发，则需要先安装 Xcode，这是一个用于 iOS 开发的 IDE。如果选择绕过 Xcode 的安装，也建议熟悉一下其相关步骤，以了解在本地平台上设置 PhoneGap 会用到的一些通用步骤。尽管每一个平台都有特定的 IDE 安装指令，但是对所有的平台来说，安装 PhoneGap、设置项目和进行部署的通用过程仍然具有一致的步骤。在设置好了 iOS 平台之后，则应该有一个类似于图 10-1 所示的 Xcode 新项目。

1 见 http: //www.phonegap.com/.
2 见 http://www.phonegap.com/about/features.
3 见 http: //phonegap.com/start#ios-x4.
4 见 http://developer.apple.com/xcode/.

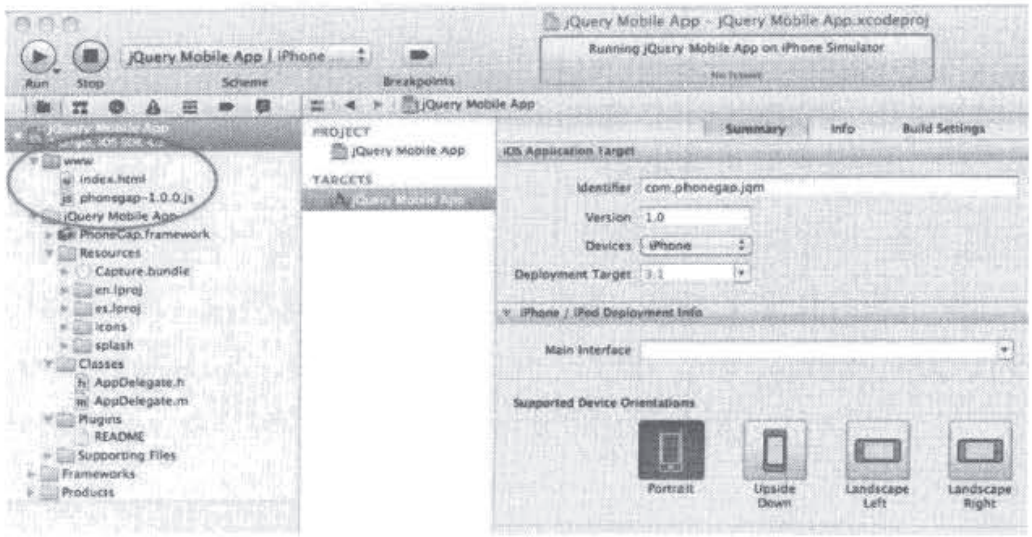


图 10–1 支持 PhoneGap 的初始 Xcode 项目

在图 10-1 中，“www”目录是应用程序的根目录。该目录内存放的是 PhoneGap JavaScrit 库和一个默认的页面（index.html）。在运行 app 时，index.html 页面是作为初始载入页面显示。在 Xcode 中，为了构建和运行 app，单击出现在 Xcode 左上角出现的“Run”按钮。在单击该按钮之后，会先编译该 app，然后启动 iOS 模拟器，然后显示 index 页面，如图 10-2 所示。

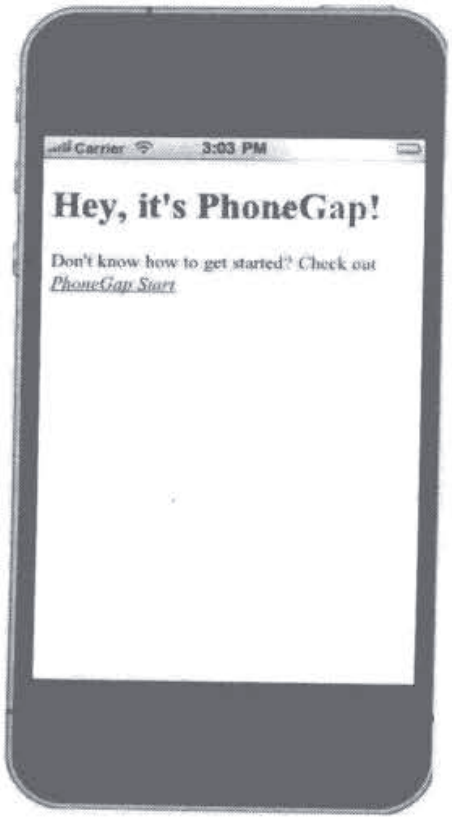
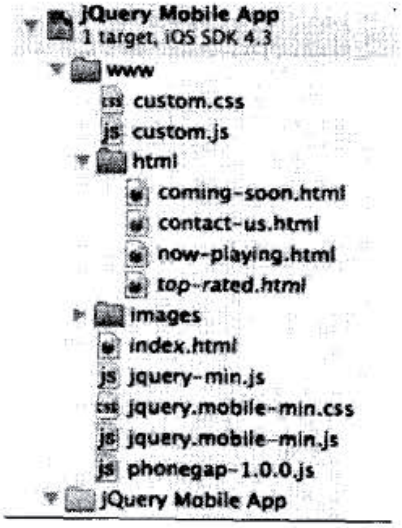


图 10–2 在 Xcode 中运行 PhoneGap 的默认 app 时，显示的初始页面

在 Xcode 中设置好了 PhoneGap 项目之后，我们可以将一个现有的 jQuery Mobile 应用程序导入到我们的项目中。将 jQuery Mobile 应用程序导入到 Xcode 项目中，以及作为一个本地 iOS 应用程序进行部署的步骤如下所示。

1. 首先，需要将一个现有的 jQuery Mobile 项目导入到 Xcode 的“www”根目录中。对该练习而言，可以导入你自己的 jQuery Mobile 应用程序，或者导入第 10 章的源代码文件夹中包含的 jQuery Mobile 项目。例如，如果我们从第 10 章的源代码目录中导入 jQuery Mobile 文件，并将其移动到我们的“www”目录，我们的 Xcode 项目结构看起来会如下图所示。



2. 在将 jQuery Mobile 项目导入到 PhoneGap 项目之后，我们需要导入 PhoneGap 的 JavaScript 库，并将其作为顶级的资源：

```
<head>
  <meta charset="utf-8">
  <title>jMovies</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" href="jquery.mobile-min.css" />
  <link rel="stylesheet" type="text/css" href="custom.css" />
  <script type="text/javascript" src="phonegap-1.0.0.js"></script>
  <script type="text/javascript" src="jquery-min.js"></script>
  <script type="text/javascript" src="custom.js"></script>
  <script type="text/javascript" src="jquery.mobile-min.js"></script>
</head>
```

PhoneGap 库是一个能够访问多个设备特定的特性（摄像头、媒体、存储等）的 API。在 PhoneGap 的站点上，有它支持的所有 API 的文档和实例[1]。只有当你的应用程序需要与 PHoneGap 的本地性能进行交互时，才有必要导入 PhoneGap 的库。

[1] 见 http ://docs.phonegap.com/.

3．最后一步是运行和测试我们的应用程序。在 Xcode 中，单击“Run”按钮。这将会编译该 app，然后在 iOS 模拟器内启动该 app。如果从第 10 章的源代码中导入了 jQuery Mobile 项目，则显示的初始界面将会是 springboard（见图 10-3）。

图 10–3 作为本地 iOS app 运行的 jQuery Mobile

为了验证 PhoneGap 的库是否正确安装，我添加了一个监听器来监听 PhoneGap 的 device-ready 事件。当该事件被启动时，PhoneGap 处于待续状态，我们就可以与 PhoneGap API 通信了（见程序清单 10-1）。

程序清单 10-1 PhoneGap 已经就绪（ch10/custom.js）

```
$(document).bind("deviceready", function(){
    navigator.notification.alert("PhoneGap is initialized...");
});
```

如图 10-3 所示，当 PhoneGap 就绪时，会显示一个警告视图，用于指示 PhoneGap 已经被初始化。在程序清单 10-1 中的警告通知就是一个我们如何以编程方式与 PhoneGap 的 API 进行交互，以访问其本地功能的示例。

> **注意：** PhoneGap 简化了将 jQuery Mobile Web app 转换为在 iOS 上运行的本地 app 的过程。从技术角度来看，我们的 jQuery Mobile Web app 现在运行在 iOS Web 视图中。

将在 Safari 浏览器中运行的 jQuery Mobile app（见图 10-4）与运行在 iOS 中的本地 app（见图 10-5）相比，你可以看到什么区别？

图 10–4 在浏览器中运行的 jQuery Mobile

图 10–5 作为一个本地 iOS app 运行的 jQuery Mobile

两者之间最明显的区别是，在 Safari 浏览器内，Chrome 浏览器是可用的，而在本地的 app 中则不可用。在第 3 章的“回退按钮”一节中讲到，在 jQuery Mobile 内，回退按钮最初是禁用的，原因是 Chrome 浏览器已经提供了内置的导航按钮。然而，对于运行本地 iOS app 的用户来说，页眉内的回退按钮是导航的一个主要手段。幸运的是，通过一个简单的配置更新，我们就可以在 jQuery Mobile 内启用回退按钮（见程序清单 10-2）。

程序清单 10-2 以全局方式启用回退按钮（ch10/custem.js）

```
$(document).bind("mobileinit", function(){
    $.mobile.page.prototype.options.addBackBtn = true;
});
```

jQuery Mobile 内的回退按钮相当智能。只有当页面历史访问记录中存在可以返回的页面时，该按钮才出现。当以全局方式启用回退按钮之后，我们的 iOS 本地用户在导航该 app 时，会感到更舒适（见图 10-6）。

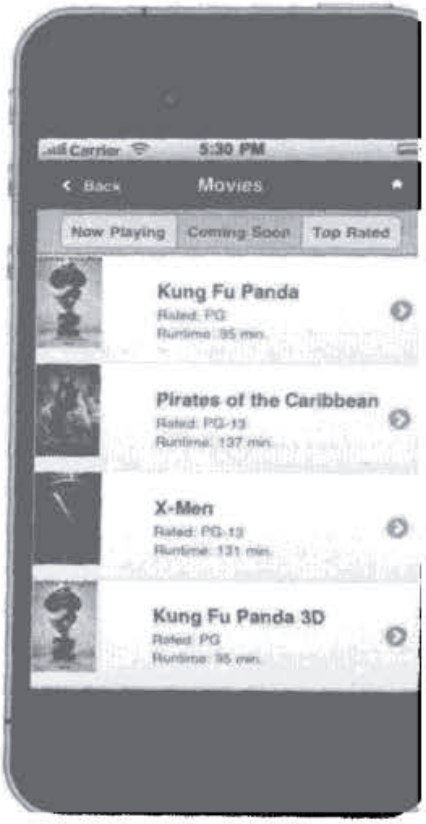


图 10-6 以全局方式启用回退按钮，以支持导航

在以全局方式启用回退按钮之后，如果在某些特定页面上不需要回退按钮时，也可以禁用它们。尤其是我们不希望让回退按钮显示在主页面上。为了阻止回退按钮在给定的页面上显示，我们可以将 data-add-back-btn=“false”属性添加到页面容器。

```
<div data-role="page" id="home" data-add-back-btn="false">
```

在进行上述设置之后，当我们导航到主页面时，则不会看到回退按钮。

我们已经可以将我们的 jQuery Mobile app 部署到本地 iOS 平台上，我们可能也想自定义默认的 app 图标（见图 10-7）和启动屏幕（见图 10-8）。

图 10-7 PhoneGap 在 iOS 中的默认图标

图 10-8 PhoneGap 在 iOS 中的默认启动屏幕

app 图标存储在项目的/Resources/icons 目录中，启动屏幕的图像存储在/Resources/splash 目录中（见图 10-9）。图像可以适用于不同的 iOS 屏幕分辨率和大小。

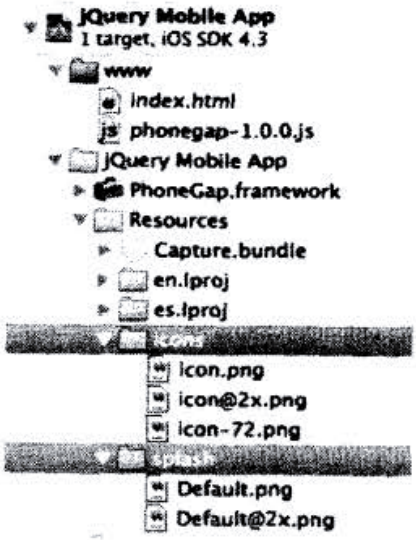


图 10-9 用于启动屏幕和 app 图标的 Xcode 图像

此外，你可以改变绑定显示名（bundle display name）和绑定标识符（bundle identifier），在 Xcode 中，这些内容可以在项目的“Info”选项卡中设置（见图 10-10）。绑定显示名设置了 app 图标的标签，而绑定标识符由 iOS 用来唯一地识别你的 app。

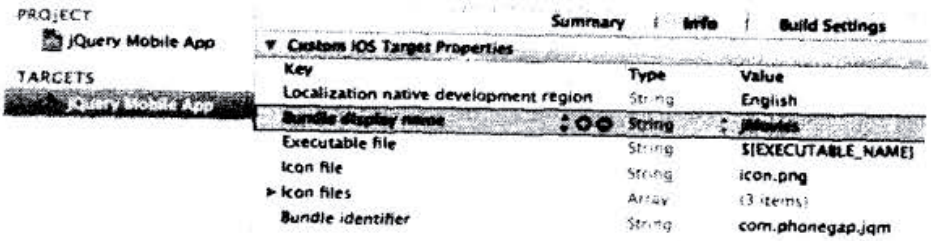


图 10-10 设置绑定显示名和绑定识别符

提示：在使用 PhoneGap 进行开发时，建议将$.mobile.allowCrossDomainPages 配置选项设置为 true:

```
$(document).bind("mobileinit", function(){
    $.mobile.allowCrossDomainPages = true;
});
```

PhoneGap 的 Web 视图允许应用程序进行跨域调用。这通常是允许的，以便应用程序能够从它的 home 服务器获取数据。默认情况下，jQuery Mobile 将跨域请求看做外部链接。因此，跨域的页面不会载入到当前页面的 DOM 中，而且也不会应

用任何转换。因此，在 PhoneGap 中，如果想允许 jQuery Mobile 管理跨域请求的页面载入逻辑，需要将该选项设置为 true。

从在本地 iOS 平台上安装 PhoneGap 到运行 jQuery Mobile app 的整个过程到此结束。当你的 app 可以作为产品发布时，最后一步是将你的 iOS app 发布到 Apple 的 App Store。尽管该过程相当冗长，但是将你的 app 发布到 Apple 的 App Store 的完整说明可以在 Apple 的 iOS developer library[1]中找到。

10.3 将 jQuery Mobile 作为一个 Android app 来运行

在本节，我们将使用 PhoneGap 包装一个 jQuery Mobile app，然后在本地 Android 平台上运行。要在 Android 平台上设置 PhoneGap，我们可以参考 PhoneGap 的“Getting Started Guide with Android”[2]。要针对 Android 平台进行开发，则需要先安装 Eclipse，这是一个用于 Android 开发的 IDE。在设置了好了 Android 平台之后，你应该有一个类似于图 10-11 的 Eclipse 新项目。

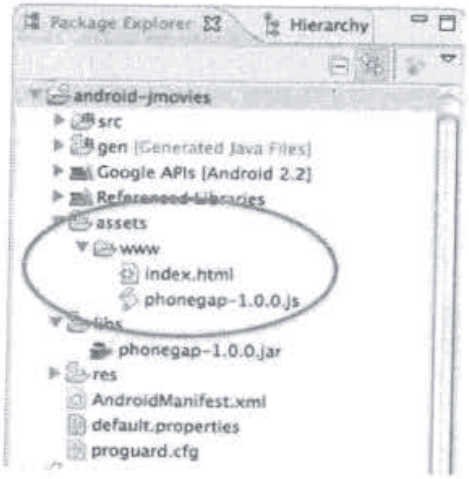


图 10-11 支持 PhoneGap 的初始 Ecplise 项目

在图 10-11 中，“www”目录是应用程序的根目录。该目录内存放的是 PhoneGap JavaScript 库和一个默认的页面（index.html）。在运行该 app 时，index.html 页面会作为初始载入页面来显示。在 Eclipse 中，为了构建和运行 app，需要单击“Run”菜单，

1 见 http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios_development_workflow/145-Distributing_Applications/distributing_applications.html.
2 见 http://www.phonegap.com/start#android.

选择“Run As”，然后选择 Android Application。在编译和运行该 app 之后，会启动 Android 模拟器，并显示 index 页面（见图 10-12）。

> **提示：**如果发现启动 Android 模拟器的时间过长，你可能更想部署在真实的设备上，以进行测试。对该步骤而言，确保你的 Android 设备启用了 USB 调试（Settings->Applications->Development），然后将设备接入系统。接下来，当你运行应用程序时，它的启动速度会快一些。

在 Eclipse 中设置好了 PhoneGap 项目之后，现在可以将现有的 jQuery Mobile app 导入到我们的项目中。将 jQuery Mobile app 导入到 Eclipse 项目，并作为本地 Android app 进行部署的步骤如下所示。

图 10-12 在 Eclipse 中运行 PhoneGap 的默认 app 时，显示的初始屏幕

1. 首先，我们需要将一个现有的 jQuery Mobile 项目导入到 Eclipse 的“www”根目录中。对该练习而言，你可以导入你自己的 jQuery Mobile app，或者是导入第 10 章的源代码文件夹中包含的 jQuery Mobile 项目。例如，如果我们从第 10 章的源代码目中导入 jQuery Mobile 文件，并将它们移动到我们的“www”目录，则 Eclipse 项目结构应该如图 10-13 所示。

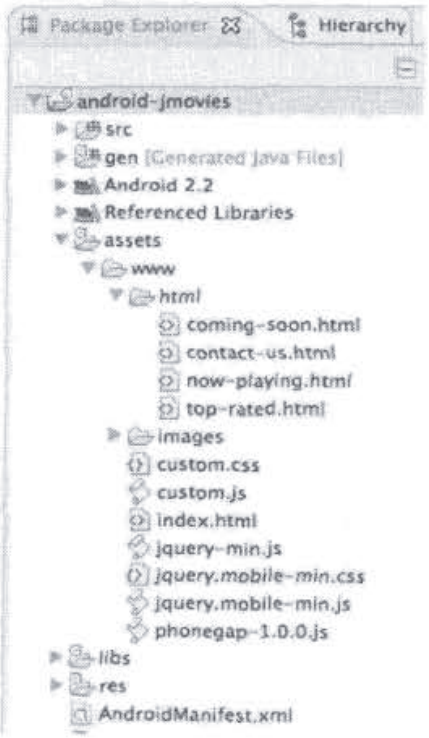


图 10–13 一个 Eclipse 项目

2. 在将 jQuery Mobile 项目导入到我们的 Eclipse 项目之后，我们需要导入 PhoneGap 的 JavaScript 库，并将其作为顶级资源：

```
<head>
  <meta charset="utf-8">
  <title>jMovies</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" href="jquery.mobile-min.css" />
  <link rel="stylesheet" type="text/css" href="custom.css" />
  <script type="text/javascript" src="phonegap-1.0.0.js"></script>
  <script type="text/javascript" src="jquery-min.js"></script>
  <script type="text/javascript" src="custom.js"></script>
  <script type="text/javascript" src="jquery.mobile-min.js"></script>
</head>
```

3. 最后一步是运行和测试我们的应用程序。在 Eclipse 中，单击 Run 菜单，选择 Run AS，然后选择 Android Application。这会编译该 app 并在 Android 模拟器内启动该程序。如果你是从第 10 章的源代码中导入的 jQuery Mobile 项目，则显示的初始屏幕应该是 springboard（见图 10-14）。

在我们前面的 iOS 示例中，我们以全局方式启用了回退按钮，以便显示在所有的界面上，这是因为回退按钮通常是固定显示在 iOS app 的页眉内部。由于 Android 具有基于硬件的回退按钮，因此没有必要启用我们的回退按钮（见图 10-15）。

图 10–14 在 Eclipse 中运行 PhoneGap 时的初始屏幕

图 10–15 Android 设备上没有回退按钮

> **提示：**将平台特定的依赖关系存放在隔离的文件中。这种隔离有助于简化针对不同设备进行配置的管理。例如，为每一个支持的平台创建一个独立的配置文件：custom-ios.js、custom-androd.js。现在每个平台都能载入它们独特的依赖关系。

现在我们能够将 jQuery Mobile app 部署到本地的 Android 平台，我们也可以自定义默认的 app 图标（见图 10-16）。

Android app 图标存储在项目的/res/drawable-*目录中，而且具有高分辨率、中等分辨率和低分辨率 3 种格式（见图 10-17）。

图 10-16 PhoneGap 在 Android 中的默认 app 图标

从在本地 Android 平台上安装 PhoneGap 到运行 jQuery Mobile app 的整个过程到此结束。当你的 app 可以作为产品发布时，最后一步是将你的 Android app 发布到 Android Market。将 app 发布到 Android Market 的完整说明可以在 Android 的开发指

南中找到[1]。

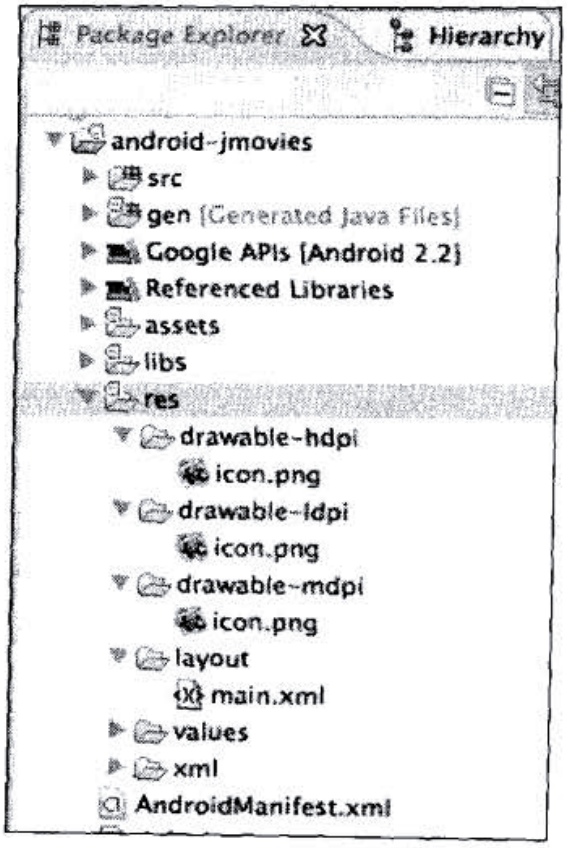


图 10–17 Android 的 app 图标

10.4 Open App Market

Open App Market[2]是一个用于 HTML5 移动 app 的 app store，它允许人们如同在本地 app store 那样搜索、购买、安装 HTML5 移动 app，以及对 HTML5 移动 app 进行打分（见图 10-18）。

为了了解 Open App Market，我们需要先将其安装到设备上。当前，Open App Market 可用于 iOS 和 Android 用户。要安装 Open App Market，需要使用你的 iOS 设备或 Android 设备扫描图 10-19 中的 QR 码。

1 见 http://developer.android.com/guide/publishing/publishing.html.
2 见 http://openappmkt.com/.

图 10-18 Open App Market

图 10-19 用于安装 Open App Market 的 QR 码。你可以访问 openappmkt.com 并单击站点顶部的安装链接进行安装

在安装 app 之后，你可以根据分类或流行度来搜索移动 Web app。当你找到感兴趣的免费或付费 app 时，可以将它下载下来，该 app 就会就保存在你的主界面中，就像是一个来自于本地 app store 的 app 那样。例如，图 10-20 显示了 Open App Market app 以及从 Open App Market 下载的 Twitter 和 YouTube app。

图 10–20 Open App Market 下载 app 的方法与本地 store 相同

10.5 客户端设备 API

如果你需要构建的移动 app 必须与设备特定的特性（比如摄像头、通讯录或者网络）相继承，则应该选择哪种移动技术呢？如今，我们的选择是有限的。我们必须使用本地平台来构建，或者是使用像 PhoneGap 这样的混合技术来构建。如果所有的 Web 浏览器也都支持这些设备特定的特性，则最理想不过。由于当今不存在支持所有这些特性的浏览器，W3C 当前正在为大多数主流的客户端设备 API[1]制定工作草案。最著名的客户端设备 API 包括对摄像头、网络、日历、通讯录、消息收发和电池信息的访问。也许现在预言浏览器何时能支持所有这些特性还为时尚早，但是人们正在朝着这个方向努力。

10.6 总结

在本章，我们讲解了如何将一个现有的 jQuery Mobile 应用程序与 PhoneGap 框架

[1] 见 http//www.w3.org/2009/dap/.

相集成。PhoneGap 为 jQuery Mobile Web 应用程序添加了两个独特的功能。第一，我们可以将现有的 jQuery Mobile Web app 包装在 PhoneGap 框架中，然后再将它发布到本地 iOS、Android、BlackBerry、WebOS 和 Symbian 平台。第二，PhoneGap 也公开了一个 API，以允许我们的移动 Web 应用程序与设备特定的 API（包括文件系统、通知、摄像头等）进行交互。PhoneGap API 允许我们以多种方式来扩展 jQuery Mobile 应用程序，而在此之前我们只能使用本地 SDK 来扩展。

我们还介绍了 Open App Market，这是一个用于 HTML5 移动 app 的 app store，它允许我们就像在本地 app store 中那样搜索、购买、安装 HTML5 移动 app，以及对其进行打分。如果人们觉得本地 app store 的发布过程相当麻烦而且缓慢的话，则可以考虑使用 Open App Market 进行替换。

最后，我们介绍了 W3C 当前正在编写的客户端设备 API。这些 API 对移动 Web 开发人员来说相当重要，因为它能够在无需依赖外部框架的情况下，允许我们的 Web 应用程序访问设备 API（日历、通讯录、摄像头等）。